Learning in neural networks: VLSI implementation strategies

Tuan A. Duong, Silvio P. Eberhardt*, Taher Daud, and Anil Thakoor
Center for Space Microelectronics Technology
Jet Propulsion Laboratory, California Institute of Technology
Pasadena, CA 91109
*Department of Engineering, Swarthmore College
Swarthmore, PA 19081

ABSTRACT:

Fully-parallel hardware neural network implementations may be applied to high-speed recognition, classification, and mapping tasks in areas such as vision, or can be used as low-cost self-contained units for tasks such as error detection in mechanical systems (e.g. autos). Learning is required not only to satisfy application requirements, but also to overcome hardware-imposed limitations such as reduced dynamic range of connections. A learning algorithm may be implemented in hardware, in which case the application merely needs to provide training data (for supervised learning), or the hardware implements only a feedforward operation, in which case learning is under the control of a host computer that applies input patterns and updates connections according to the error of the measured outputs (i.e. hard ware-in-the-loop learning). The latter method is useful if the network only needs to be trained once for an application, since it greatly simplifies the hardware, but at the cost of greater learning time and the requirement for a host computer.

Following a review of the emerging hardware implementation strategies for neural network learning reported in the literature, this chapter details a new architecture and supervised learning algorithm, cascade backpropagation (CBP). It combines powerful features from other algorithms such as cascade correlation (CC) and error backpropagation (EBP), and is particularly suited to problems of image/data classification and object discrimination. CBP is a constructive architecture in which a neuron (processing unit) is sequentially added to the net, and gradient descent is used to permanently fix the weights connected to that neuron, both input and output. Each new neuron has connections to the inputs and each preceding neuron's output; thus each added neuron implements a hidden layer. The addition of each successive neuron provides the system with an opportunity to further reduce mean-squared error. Because the average number of connections to a neuron is small, learning is quite fast.

Currently, the system is implemented using analog CMOS VLSI and hardware-in-the- loop learning. To adapt the architecture for hardware with limited synaptic dynamic range, the maximum synaptic conductivity associated with later neurons is reduced, thus effectively reducing the synaptic quantized step size. Simulations and tests with analog CMOS VLSI hardware suggest that the system is capable of learning difficult problems (such as 6-input panty and image classification) with synaptic quantizations as low as 5 bits, as opposed to the 8-16 bits required for EBP and CC learning algorithms.

1. INPRODUCTION

Modern general-purpose computers allow simulation of almost any neural network architecture and learning algorithm, and there is little doubt that such simulations in many cases afford the easiest and most cost-effective approach for neural network applications. However, there are major application areas that require or benefit fı om custom neural hardware. Custom hardware is necessary in cases where required throughput is greater than can be sustained on available computers, due either to very large network size or the need for short (realtime) learning or response intervals. At the other extreme, a simple network used in a mass-produced commodity such as an automobile may only be cost-effective as a single-chip standalone neural system. In between, one might envision many applications where inexpensive, self-contained “black box” neural hardware perform tasks such as fault detection, actuator control, and adaptive home environmental regulation, to name just a few.

The field of neural network hardware is still in its infancy. Perhaps the only well-established neural hardware is the computer plug-in “accelerator board" that is capable of rapidly calculating such common neural primitives as multiply-accumulate, while other computational tasks are performed by the main processor. Despite the fact that dedicated neural chipsets have been on the market for close to a decade[1,2], few custom neural systems have found their way into commercial products[3,4]. In this chapter we review the most common technologies and techniques for implementing custom neural systems, and give an example from our own work of an analog neural system capable of learning. We also stress the point that many of the learning algorithms that have been developed for computer-based simulations arc not directly compatible with hardware, and so any hardware development effort must have as it’s centerpiece the development of a compatible learning algorithm.

The basic task of a neural network hardware (or hardware simulator) is independent of the implementation technology, and can be divided into the following modes: 1) MAPPING a specified neural architecture onto the hardware taking stock of the number of inputs and outputs, etc.; 2) LEARNING: Calculation and programming of synaptic connections (and possible network architecture) so that the network will perform desired mappings from input to output; and 3) OPERATION: Upon application of a complete set of inputs, the hardware must provide the results for the required input to output mapping.

Let us consider the computational load required by learning, evaluation, and operation. For serial simulations, the following algorithm applies to many learning methodologies:

```
1. while (network hasn’t reached desired performance level)
{
        2. for (each training vector)
        {
                apply input vector
                3. for(each layer)
                {
                        4. for(each neuron in layer)
                        {
                                5. for(each synapse connecting to neuron)
                                {
                                        multiply activation by synapse weight
                                        accumulate result into neuron's input
                                }
                                calculate neuron's activation
                        }
                }
                read outputs
                calculate error at output
                perform weight updates
        }
}.
```

Note that the weight update sequence may require many more computations than the operation pass(loops 3-5), depending on learning algorithm. Due to the nested nature of the loops, computational load can increase dramatically with increased problem complexity (giving more Loop 1 passes), increased network size (increasing Loop 3,4, and 5 passes), or complex learning algorithm (increased computation in the inner loops). For example, a 2-2-1 network requiring, say, 100 passes to learn the exclusive-OR problem (2-bit parity) would require on the order of

(100 training passes) x(4 training patterns) x(2 passes) x(3 layers) x(2 neurons/layer) x (2 synapses/neuron) = 9600 operations

where it is assumed that weight-update and operation passes have equivalent complexity (2 passes). Applying this simplistic model to 3-, 5- and 8-bit parity, assuming that the number of hidden units is the same as the number of input layer units and the required number of training passes increases to 500, 2000 and 10,000, we see that the required number of operations are, respectively, 216K, 9.6M, and 983M. It is obvious that larger applications would benefit tremendous] y if calculations could be carried out in a parallel fashion.

Nevertheless, the general-purpose computer simulation that carries out these calculations one at a time is near-ideal in all respects save response time and, in some cases, cost. The computer can be programmed to implement almost any architecture and any learning algorithm, and signals are represented with high' precision and dynamic range by floating-point variables. The same is not true of custom hardware: networks are often limited in size, support only certain classes of architecture, represent signals and weights with limited precision and (possibly) with large noise components. Thus, the success of a hardware implementation depends critically upon a judicious balance of the many tradeoffs, involved in selecting a hardware technology, designing the circuits, fixing classes and sizes of architectures that will be supported, and crafting a compatible learning algorithm.

While we focus in this chapter on the most popular implementation technologies, namely, complementary metal oxide semiconductor (CMOS) analog and digital, with signals that are continuous-valued (or quantized to many levels), we should mention several other technologies that have been reported in literature for neural hardware implementations. Optical[5], thin film[6-9], and charge-coupled device [10, 11] technologies have all been used to implement neural networks, but require rather elaborate or specialized fabrication processes. A technology that uses standard CMOS, and that more closely models biological neural "wetware" functions, is pulse-mode circuits. Pulse-mode networks represent signals by the duty cycle or firing rate of pulse trains [12- 15]. Weights, however, are generally stored using analog or digital memories. A primary advantage of pulse-mode circuits lies in their space-efficient processing circuits, which combine analog circuit characteristics such as few transistors per processing element and fully-parallel implementations with small-sized transistors. A possible advantage- and the primary disadvantage- of this approach is that dynamic range is traded off with time. Otherwise, pulse-mode circuits tend to exhibit the same difficulties as analog networks [16]. Several smaller-scale pulse-mode networks have been built and furnished with a learning algorithm [13,15].

In the following sections, we hope to give the reader an appreciation of the characteristics of the more dominant technologies employed for neural implementations, including strengths and weaknesses that dictate the form of the implementations. Since our discussion of learning requires addressing details of hardware implementations, hardware is treated first. Issues having to do with learning, and in particular the incompatibility of many learning algorithms with limited-precision hardware, arc then discussed. We present in detail a new learning algorithm, CBP, that is compatible with hardware implementations, and give results from learning experiments conducted with CBP. In addition, wc describe a further refinement in the learning algorithm and give simulation results to show that the method is particularly useful for reduced weight resolution, and therefore, suitable for analog hardware implementations.

2. HARDWARE OVERVIEW

One key characteristic that distinguishes different implementations is the level of parallelism. A neural net program running on a personal computer or workstation has no parallelism beyond engaging integer and floating-point processor units simultaneously. At any point in time, the general-purpose processor is calculating one synaptic weighting, the activation function of one neuron, or one connection-update. Many custom digital implementations (and also software implementations executing on a parallel computer system) follow this model in a semi-parallel way: each processing element calculates a subset of the network's connections and neuron activations. Throughput is increased by sharing the task over multiple processors. Further gains may be achieved by designing specialized custom processors optimized for the neural tasks required. However, there is often a tradeoff between speed of execution, network size, and generality, so that the more specialized processors are usually faster, but support only a limited number of architectures and learning algorithms.

The highest level of parallelism is achieved by implementing each synapse and each neuron as a distinct circuit. While fully-parallel digital networks are rare in digital implementations, due to the silicon-hogging nature of digital weighting, aggregation, and activation-function lookup circuits, full-parallelism is the rule in analog and other non-digital implementations, in part because multiplexing is more expensive, noise-prone, and difficult in the analog domain. Also, custom circuits that store weights, perform weighting, and implement neurons tend to be significantly smaller when implemented as analog rather than digital circuits, because the physics of semiconductor circuits can be exploited to obtain neural functionalities in a highly space- and power-efficient manner [17, 18].

Because both analog and digital technologies have their own particular inherent advantages, neither has yet come to dominate. The state-of-the-art of hardware implementations can be abstracted from Table 1, which presents an overview of many of the hardware implementations that have been prototype to date. For each design, the table includes, if available in the literature, specifications such as architecture size, speed, and learning algorithm supported. Most of the features listed in the table will be discussed in the following sections[19].

Table 1. A comprehensive survey and compilation of the hardware implementations of neural networks reported in literature[19].

3. DIGITAL IMPLEMENTATIONS

Digital circuits, because of their high noise immunity and the resulting capacity to transfer information perfectly intact, arc well-suited to time-multiplexing. Partially-parallel systems, with a handful of custom synapse and neuron circuits can be constructed, allowing a flexibility that is difficult to achieve in analog technology: a given neuron circuit or synapsc multiplier can calculate one, ten or even thousands of neurons or connections within one pass with parallel processing. Thus, as long as sufficient memory for weight storage is available, almost any size neural architecture may be mapped onto a fixed number of processors. Bandwidth, often measured in millions of connections per second (MCPS) for data pıocessing operation, and millions of connection updates per second (MCUS) for learning, remains relatively constant for such a circuit. Thus, the operation throughput, the reciprocal of the time lag between application of an input and availability of the output, is also inversely related to the complexity of the architecture that is mapped onto such a time-multiplexed system.

Most digital designs are indeed multiplexed, for the simple reason that a fully-parallel digital system is impractical for all but small architectures, since each connection in the network generally requires silicon-hogging multiplier and adder circuits. Also, it would be wasteful to incorporate into each neuron circuit an activation-function lookup table, when the speed of the system is limited not by the lookup time, but by the multiply-accumulate synaptic weighting and aggregation calculations. However, a moderately-large fully-parallel digital system, described below, has been constructed using eight 5-inch silicon wafers [20].

Another advantage is that digital circuits arc scalable --- as fabrication technologies (which are often geared towards digital requirements) improve, circuits can bc made ever smaller, allowing more processors on a chip, while simultaneously decreasing execution time. Analog circuitry may not scale down to the same degree, because noise may increase as feature size is reduced. For example, transistor edge effects may become more pronounced as transistors are made smaller, resulting in larger voltage offsets. Also, electromagnetic pickup from adjacent wires may increase as wires are deposited closer to each other, and larger temperature variations may occur as current densities and fluctuations increase. As noise levels increase, it can be expected that noise-intolerant learning algorithms will fail, and that noise-tolerant algorithms will take longer to learn. However, the analog vs digital trade-off will still be decided based on type of application, requirement of precision, power consumption, processing time, and silicon real estate.

Recent innovations in digital design and fabrication technologies have already been applied to neural networks, making possible several of the larger-scale designs in Table 1. Wafer-scale integration (WSI) has been used to implement a single integrated circuit on a 5-inch diameter wafer, implementing 576 time-multiplexed neuron circuits with associated synapses, using 40 million transistors [21]. The wafer was mapped with a fully-connected network implementing, without learning, the 16-city traveling salesman problem. A solution was obtained after only 0.1 s., giving a throughput of 1.2 GCPS (gigs connections per second). An even more ambitious implementation involved a battery of eight 5-inch wafers as a WSI neural network with 1000 neurons [20]. EBP learning was incorporated into the hardware, giving a maximum weight-update rate (with all neurons used) of 2 GCUPS. The system was initially used for signature verification and stock price prediction, and it was found that the hai dware learned 1-3 times as fast as a simulation running on a Hitachi S-820 supercomputer. While these two designs illustrate how rapidly new digital technologies can be adapted to neural networks, note that WSI systems would be quite expensive, even if mass produced .

4. ANALOG IMPLEMENTATIONS

Given the "messy" analog electronic circuit milieu of time-invariant offset voltages and currents, induced and generated noise, drift, and temperature dependence, it is rather remarkable that analog networks can be made to function at all! However, biological "wetware" is at least as noisy a medium, and certainly nature has found a way to overcome the noise effects sufficiently that neural systems can exhibit highly discriminatory behavior. Carver Mead has argued that the physical characteristics of (subthreshold) analog circuits model closely those of biological neural tissue [18]. This suggests that once we know the learning algorithms employed by biological neural systems, wc may be able to directly apply these to analog hardware. Moreover, even with current analog hardware it appears that collection of noisy and imprecise neurons and synapses can behave with much higher accuracy than can the individual components [22]. Thus, the challenge is to find the architectures and algorithms that best learn a task, while reducing the effects of the underlying circuit nonidealities. However, investigators arc also attempting to "clean up" analog circuits by reducing noise components, in one case by automatically canceling offset voltages [23].

Most analog neural implementations developed so far show marked similarities (Fig. 1). Inputs, coded as voltages to take advantage of a wire's ability to distribute voltage, are routed to one multiplier circuit for each synaptic connection. Each multiplier derives the other input from a memory cell in which is stored that synapse's weight. Multiplier output signals, representing the weight inputs for the next layer of neurons, are coded as currents to take advantage of a wire's ability to aggregate currents. Finally, neuron circuits apply a nonlinear activation function to the aggregated weighted inputs, and supply a voltage. output that can be routed to the next layer of multipliers, or used as system output.

From this description, it can be inferred that a primary advantage of analog implementations is that the aggregation operation is essentially performed by the interconnect wiring, whereas in digital implementations, each synapse output must be aggregated to the appropriate neuron input net using one adder time-slot. Another advantage of many analog implementations is that the dynamic range of analog circuitry is limited by the noise characteristics, not the number of bits, as in digital circuits. Thus, whereas a semicustom digital network may be limited to fixed 8-bit weight resolution by the designers, analog networks may reach 10 bits of resolution, with stochastic (noise) effects that may additionally serve to mitigate some of the problems associated with hard-quantized networks and learning. For example, in cases where a neuron is clipped during learning, a digital weight-update signal to the synapses feeding that neuron may always be zero, whereas an analog update signal with noise may succeed in dislodging the neuron, thus bringing the network out of a local minimum.

Several primary differences beyond mere circuit detail serve to distinguish implementations, and have major effects on system performance and capabilities. First and foremost is network architecture. The architecture can either be fixed, in which case only subsets of that particular hardwired architecture may be mapped onto the chip, or programming circuitry can be added to allow some flexibility in routing synapses to neurons and perhaps controlling the number of layers in the net. Because such programming circuitry can take up a significant part of a chip's silicon real estate, the total number of usable synapses and neurons pcr unit silicon area will be correspondingly lower. The most general architecture is the fully-connected recurrent network, in which each neuron's output is routed through synapses to each neuron's input. By setting all feedback synapses to zero, any feedforward network can be mapped onto this architecture. While the network is very general, at least half the synapses are unused in a feedforward network, and likely many more, since it appears that many synapses are unnecessary even in feedforward networks [24].

A paradigm that has proven popular is the building-block approach [25] , where several chips can be interconnected in different ways to obtain a good measure of architectural flexibility. For example, synapse arrays can be implemented on independent chips, or each chip could implement onc layer of a feedforward network. Large networks could be constructed by tying many chips together. Disadvantages of this approach are the added chip and interchip wiring costs, and the throughput penalties resulting from the capacitances associated with chip pins.

A second difference lies in the method used to store connection weights. Currently, three mechanisms are being exploited: digital memory with analog converter, capacitive charge retention, and floating gate. While digital weight storage does not have the drift problems of the other methods and allows fast downloading from computer, it is space-intensive. since a digital-to-analog converter (DAC) must be furnished on-chip for each weight, and resolution is limited to about 6 bits [24,26]. Storing weights as charge on capacitors [27] is relatively space-efficient, but requires that leaked-off charge be periodically replaced, either by interspersing learning cycles with feedforward passes [28,29], or by storing weights digitally and sequential y refreshing the charge using one or more off-chip DACS [30]. Finally, floating-gate memories [1,31] store charge with non-volatility using electrically erasable programmable read only memory (EEPROM) technolog y. The charge can be non-destructively and continuously monitored by special transistor structures. Charges are added or removed by quantum-mechanical tunneling, a process that is slow and may require voltages that over time damage the storage device. Nevertheless, floating-gate memories hold much promise for single-chip standalone neural networks with on-chip learning, particularly where fast learning is not required.

Finally, the last major factor distinguishing analog learning implementations is support for learning. On-chip learning support ranges from none, as is the case with feedforward hardware networks where learning is executed solely by computer, to sophisticated stand-alone systems where the learning portions of the chip may far exceed in size the feedforward execution portions. This topic will be pursued extensively in a later section.

Let us consider in greater depth a typical analog implementation. The majority of analog designs to date have used the charge-storage mechanism for weight memories, with a sample-and-hold gate controlled by select logic, as shown in Figure 2. External address lines are decoded to allow random access of a synapse. This closes the sample-and-hold for that synapse, and allows the voltage from external computer-controlled digital-to-analog converters (DAC) to be applied to a capacitor that stores the weight. (Thus, strictly speaking, the weights are actually stored in the computer's memory in high-resolution binary form, and the capacitors just serve as a temporary store.) The alternate capacitive memory meehanism, which is employed by networks that must periodically learn in order to refresh weights, employs circuits that add or extract a quantum of charge from the capacitor [32]. In either case, the weights are applied to a multiplier circuit that continually performs the weighting function. Unfortunately, multiplier circuits require a number of sizable transistors if linearity, uniformity between circuits, and a large dynamic range are desired, limiting synaptic circuit density to the order of 10^3-10^4 synapses pcr cm^2. Since number of synapses is the limiting factor determining the size of most fully-parallel networks, the usc of simpler nonlinear multipliers has been proposed, along with a learning scheme tolerant of the nonlinearities [33]. Fairly linear response has also been obtained using simpler weighting circuits [34]. Synaptic outputs, coded as currents, are simply and highly efficiently aggregated by the wires that connect them to their corresponding neuron input. Analog semiconductor physics can also bestow an advantage in the neuron circuit --- a sigmoidal activation function can be efficiently implemented by little more than a differential transistor pair.

Let us highlight just two of the significant numbei of analog implementations that have been developed. (Additional citations are given in the section on learning, below). One AT&T neural system is particularly interesting because it is an analog-digital hybrid that is being used for character recognition [35]. With a recognition rate of onc hand written character every millisecond, thc system was faster by two orders of magnitude than a (serial) digital signal processor (DSP) optimized for neural calculations. A 20x20 pixel image was applied to 4 network layers mapped onto an analog chip that implements 130,000 connections, and a final 5th layer was implemented serially on a DSP using an additional 3000 connections. Learning was performed off-line by a workstation, and weights were downloaded to the system's memories. The analog network used the capacitor-charge method for buffering weights. Weights were quantized to 6 bits, and neuron activations were represented with only 3 bits, including sign, The relatively few quantization levels necessitated a final learning stcp where the weights in the final layer were retrained on-line. The recognition error rate was 5.3%, as compared to rates of 4.9% and 2.5% for full dynamic-range simulations and human subjects, respectively.

Onc of the earlier commercially-available products was the Intel's electrically trainable analog neural network (ETANN) chip, released in 1990 [36]. Each ETANN 80170W chip comprises 64 neurons and 10240 floating-gate synapses onto which can bc mapped recurrent or multi-layer feedforward networks. The chip must be plugged into a socket on a development system for learning, using onc of the many supported learning algorithms. While learning is slow duc to the floating-gate technology, ETANN has nevertheless heralded the age of program-rarely, moderately-sized, standalone analog neural network chips.

4.1 JPL Hardware Approach

At the Jet Propulsion Laboratory, we have developed a variety of "building-block" chips, some of which use digital weight storage, and some of which use capacitor storage [26,30]. Learning has been demonstrated with both these designs [24,37]. In this chapter, we highlight one of the chipsets, which incorporate hybrid multiplying digital-analog convertor (MDAC) synapses, on which we have implemented learning. Each synapse (Fig. 3) consists of a 7-bit digital memory that can be randomly accessed by a host computer, a 6-bit digital-to-analog converter using scaled current mirrors, a circuit to convert the input voltage to a current in order to drive the converter's current mirror network, and a programmable current-steering network such that the synapse can be programmed to be excitatory or inhibitory. Each synapse circuit is 200x200μm in a 2μm CMOS fabrication process.

The neuron circuit is slightly more complex (Fig. 4). To avoid a speed penalty resulting from having to charge and discharge large summing-node capacitances (especially if the.se nodes are routed between chips), the potential of each current-summing "net" node is held constant by the corresponding neuron circuit. This is achieved by the neuron's input stage: a differential transistor pair Q19-Q20 amplifies the deviation of the summing node from ground (i.e. half the 10V power supply potential), and causes the generation of a current that opposes the sum current, forcing the potential of the sum node to remain at ground. This compensating current is mirrored, inverted, and applied to an output transimpedance node. The transimpedance, and thus the neuron's sigmoidal slope (i.e. gain), can be controlled over a wide range by a programmable current mirror circuit (Q 14). Programmable neuron gain is useful for normalizing the neuron's response for the number of input synapse connections [30,38]. This design resulted in a wide range, variable gain neuron.

These circuits were combined on two chips with two types of architectures. One type implements a 32x32 crossbar network of 1024 synapses; the other is similar except that the main diagonal consists of neuron circuits. These two types of chips can be cascaded and programmed to form larger, fully -connected, partially-connected, or feedforward layered networks. A variety of network architectures (with standard synapse and neuron characteristics) can be constructed with this chipset. To map a feedforward network onto a chipset that is wired to be fully-connected, all synapses leading to a previous layer are simply nulled. Respective synapses on two chips can even be paralleled together to increase the number of synaptic quantization steps[38]; the outputs of the two synapses are wired in parallel, and the synapses on the chip with the most-significant bits are provided with 64 times the transconductance of the respective synapses with less significant bits.

4.1 JPL Hardware Approach

At the Jet Propulsion Laboratory, we have developed a variety of "building-block" chips, some of which use digital weight storage, and some of which use capacitor storage [26,30]. Learning has been demonstrated with both these designs [24,37]. In this chapter, we highlight one of the chipsets, which incorporate hybrid multiplying digital-analog convertor (MDAC) synapses, on which we have implemented learning . Each synapse (Fig. 3) consists of a 7-bit digital memory that can be randomly accessed by a host computer, a 6-bit digital-to-analog converter using scaled current mirrors, a circuit to convert the input voltage to a current in order to drive the converter's current mirror network, and a programmable cur-rent-stem-kg network such that the synapse can be programmed to be excitatory or inhibitory. Each synapse circuit is 200x200μm in a 2μm CMOS fabrication process.

The neuron circuit is slightly more complex (Fig. 4). To avoid a speed penalty resulting from having to charge and discharge large summing-node capacitances (especially if these nodes are routed between chips), the potential of each current-summing "net" node is held constant by the corresponding neuron circuit. This is achieved by the neuron's input stage: a differential transistor pair Q19-Q20 amplifies the deviation of the summing node from ground (i.e. half the 10V power supply potential), and causes the generation of a current that opposes the sum current, forcing the potential of the sum node to remain at ground. This compensating current is mirrored, inverted, and applied to an output transimpedance node, The transimpedance, and thus the neuron's sigmoidal slope (i.e. gain), can be controlled over a wide range by a programmable current mirror circuit (Q) 4). Programmable neuron gain is useful for normalizing the neuron's response for the number of input synapse connections [30,38]. This design resulted in a wide range, variable gain neuron.

These circuits were combined on two chips with two types of architectures. One type implements a 32x32 crossbar network of 1024 synapses; the other is similar except that the main diagonal consists of neuron circuits. These two types of chips can be cascaded and programmed to form larger, fully-connected, partially-connected, or feedforward layered networks. A variety of network architectures (with standard synapse and neuron characteristics) can be constructed with this chipset. To map a feedforward network onto a chipset that is wired to be fully-connected, all synapses leading to a previous layer are simply nulled. Respective synapses on two chips can even be paralleled together to increase the number of synaptic quantization steps[38]; the outputs of the two synapses are wired in parallel, and the synapses on the chip with the most-significant bits are provided with 64 times the transconductance of the respective synapses with less significant bits.

Sign bits are programmed together. While the response of such stacked synapses may not increase monotonically with binary weight count, it is advantageous with some learning schemes to have the additional levels of weight quantization.

4.2 A Technology Of The Future: 3-D Die Stacking*

JPL is currently evaluating an approach that may allow the construction of very large analog or analog/digital neural systems. Noting that size of the VLSI networks is often limited by available silicon area (where area, in turn, is constrained by increasing cost and decreasing reliability as die size increases), the possibility exists that functioning silicon dies can be interconnected by stacking to form compact, three-dimensional structures. A cube, constructed from scores of thinned dies, occupies approximately the. same footprint as a standard die. In addition to the tremendous processing power afforded by such a dense integrated circuit (IC) cube, hybridization of a 3-D IC stack to an image sensor array would enable spatially parallel signal processing to be performed on image data at extremely high data rates. As shown in Figure 5, an architecture has been conceptualized which combines the spatially parallel 3-D imager cube with neural network processing for the first time, promising tremendous speed and network size enhancements over conventional 2-D VLSI techniques[39]. While the feasibility such stacking technologies has been demonstrated [40], many challenges must be faced in developing such a cube, including heat control, the development of software tools that can follow connections in the third dimension, and, of course, the development of an appropriate neural-based architecture.

A particularly challenging application that requires the tremendous processing capability afforded by such a 3D neural image processing cube is missile defense, which specifies spatial-temporal recognition of both point and resolved targets at extremely high speed (milliseconds). A reconfigurable neural network architecture, properly trained, may discriminate targets from clutter or classify targets once resolved. By mating a 64 x 64 image sensor to a stack of 64 neural net ICS so that each row in the imager array is attached to one IC, each with a different set of weights, a variety of image processing tasks could be performed in parallel at extremely high speeds and in an extremely small package. Neural network inputs could be controlled by a sequencer circuit that controls signal flow along 64 common bus lines. A novel sequencer circuit comprises a switching matrix that allows a small window (e.g. 8 x 8) from the imager to be input to any IC in the stack.

In order to limit power dissipation to about 2 watts for the entire IC stack, the synapse and neuron circuits described above were redesigned to support lower operating currents and power supply rails, and a concomitant four-fold speed increase. 1 'he expected computation rate for a 64-

die stack incorporating these synapses and neurons would be 10^{12} connections per second, and could be increased to 10^{18} CPS when a 1024x1024 focal-plane array imager becomes available and as the 3-D stacking technology matures further. The synapse circuit is similar to the earlier version, except that it utilizes single transistor current mirrors rather than the cascode current mirrors of the previous design. The neuron circuit, shown in Fig. 6, consists of a very simple variable gain transconductance operational amplifier without compensation capacitor. Neuron gain is varied by adjusting the amplifier bias current. Figure 7 shows the two-quadrant synapse output characteristics of the hardware as a function of stored weights with the voltage V_{in} as a parameter. The combined synapse-neuron characteristics are shown in Figure 8 as a family of sigmoidal curves with different slopes obtained by variation of the gain voltage. These circuits were modeled with a PSPICE circuit simulation tool and experimental results correlate closely with simulation. Simulation results indicate an average power consumption of less than 30 milliwatts/chip (or less than 2W for a 64-chip stack) at the 4 MHz operation rate.

5. LEARNING IN HARDWARE SYSTEMS

A general-purpose computer can be programmed to execute any reasonably-sized architecture and any conceivable learning algorithm. The dynamic range of weights and signals traversing the simulated network, coded with floating-point variables, is sufficiently large that quantization effects very rarely affect learning or operation. Unfortunately, the opposite is true of most analog or digital hardware implementations: signals and/or weights must be implemented with limited quantized levels of resolution and dynamic range. Studies suggest that for most learning algorithms a reduced dynamic range will adversely affect (or even inhibit) learning. For operation, however, reducing the dynamic range of weights and signals to a few bits often does not greatly affect the result [17,41]. A direct implementation of the ever-popular EBP algorithm, for example, requires 12-16 bits of weight quantization [42]. However, major modifications of EBP may function reliably for at least some problems with as few as 8 bits of weight precision [43].

Learning with analog hardware poses a second challenge: how to structure a learning algorithm to be less sensitive to the noise sources inherent in analog circuits Such sources can be dynamic, with wide-ranging frequency components (including low-frequency drifts), or time-invariant, as in the fixed offset signals generated within every analog circuit. Furthermore, noise sources are not necessarily uncorrelated: noise in power busses may affect circuit outputs in diverse ways. As mentioned above, noise can in some cases assist learning by introducing a stochastic component to weight updates. However, offsets can be a major problem, as can correlated noise sources.

Thus, a primary challenge that faces the hardware designer is finding a hardware-compatible learning algorithm. We will focus here on a few leading examples of supervised learning algorithms that appear to be most promising for hardware learning[44]. For the many applications that do not require fast learning (including situations where the weights are fixed for the life of the network), learning may be under the control of a computer. Digital networks of modest size can often be faithfully simulated using floating point variables, and the resulting weights can be quantized and mapped onto the hardware. Such an approach may not work for analog networks unless offsets and other noise sources are measured and incorporated into the simulation. Instead, a simple but time-intensive gradient-descent method that has been employed is hardware-in-the-loop learning (HILL) [28,45-47]. HILL systems use a computer to set the analog input values to the hardware, measure the outputs, and reprogram weights. A training token is applied to the network, and the output is compared to the target vector. Each neuron output or weight is in turn perturbed, and the effect of the weight perturbation on the output error is calculated. The weight is then modified slightly so as to decrease the error. Obviously, this is a highly inefficient learning method, even if several simultaneous weight updates can be made at once. Nevertheless, the advantage of this scheme is that all time-invariant noise and other nonideal hardware behavior is taken into account, including even altogether malfunctioning circuits.

Other investigators have included circuits for learning on-chip. A computer may still be necessary to apply training vectors, but learning can usual 1 y proceed much more rapid] y due to higher weight-update parallelism, and faster learning cycles. While many investigators have designed and even partially implemented analog networks with on-chip learning, using supervised learning algorithms such as EBP[29,48-52] and other gradient-descent techniques [53], or unsupervised learning algorithms such as Oja's rule [54] and Kohonen networks [55-57], relatively few functional analog on-chip learning systems (beyond limited prototypes) have been reported. Pioneering experiments with small networks capable of learning were pursued starting in the 50's by Widrow [58], using his madaline learning mechanisms. Alspector has successfully executed several designs, using (stochastic) Boltzmann Laming [59]. His more recent stochastic system used controlled noise sources in the form of digital circuits that generated random bitstreams with low correlation [60,61], rather than the uncontrolled sources inherent in the analog circuitry. Finally, a more specialized analog implementation used Grossberg self-organized learning [62]. Digital on-chip learning networks have also been implemented, generally with EBP learning variants [3,19,63]. Onc noteworthy neural chip with a measured time for a feedforward pass of only 104 ns used a variant of restricted-Coulomb energy (RCE) learning [64].

6. SELF-EVOLVING ARCHITECTURES

Most learning algorithms operate on a fixed architecture that has been predetermined, often using little more than guesswork. The problem is that the network size required for a given problem is dependent on the complexity of the input data set and the structure of the patterns to be extracted. These factors are generally unknown. If the selected architecture is larger than required for a particular problem, learning may take longer than necessary, and if the selected architecture is too small, the network will not adequately learn the task at hand. To avoid the necessity of fixing a network architecture, and to obtain higher efficiency in learning, a new class of learning architecture has been proposed in which a network evolves out of a si mplc two-layer precursor architecture. Hidden units are added as necessary until the network performs adequately.

The first such architecture appears to be Scott Fahlman's CC learning scheme [65]. The precursor network has no hidden units, and weights are adjusted using the gradient descent (or one of its variants). Then, in each subsequent operation, a new single-neuron layer is added, with the neuron's inputs connecting to the network inputs as well as all hidden-unit neuron outputs. Initially, a new neuron's output is not connected, and the input weights arc set so as to maximize the covariance between the new neuron's output and the residual error of the network output. These input weights arc not altered after this. Finally, all output-layer weights are retrained using the delta rule. In this way, each new neuron serves as a feature detector that is likely to reduce the output error, and which can be used by subsequent neurons for more sophisticated features. A final advantage is that the rate of decrease of error with each new hidden-unit addition can serve to indicate the utility of adding further units.

Such an architecture has a number of attractive features for usc with hardware implementations. Besides the advantages deriving, from architectural efficiency, such as efficient network size and use of a small network for at least part of the training task, each of the two steps of the learning algorithm requires updating relatively few synapses. Furthermore, an error signal does not need to bc propagated back across multiple layers - a process that is highly noise-prone in analog implementations.

A study of the sensitivity of CC learning to reduced dynamic range variables and weights has shown that while the algorithm is relatively insensitive to representing neuron activation by even as few as 5 bits of precision, weights must be represented with a much greater dynamic range [41]. The 6-bit parity was one among various problems studied in simulation, where the limited weight precision led first to an increase in network size, then catastrophic failure below about 12 bits as weight updates were mostly truncated to zero. Modifications of the algorithm that included probabilistic weight update resulted in successful learning, with as few as 7 bits [41]. However, these modifications would be expensive to implement in hardware.

6.1 Cascade Backpropagation (CBP) Learning Architecture

In this section we develop a new self-evolving architecture that is highly efficient with respect to hardware implementations, and demonstrate its ability to learn with reduced synaptic weight dynamic-range. This new learning architecture is termed CBP and it is shown in Figure 9. In comparison with EBP, CBP was designed with a clear motivation to avoid the arbitrary and predetermined assignment of hidden units, and thus avoid identical subspaces in weight-space that may cause convergence problems[66]. In addition, its most important feature is the capability to reduce the weight resolution requirement of EBP, which is particularly costly to implement in hardware [42]. Further, the theory of self-evolving architecture shows that each added hidden unit potentially reduces the energy level, which continuously moves the network towards minimum energy level [67].

CBP uses the stochastic gradient descent technique and the self-evolving architecture [65]. The process of adding a new hidden unit is based on a number of fixed iterations. Learning is required for the synaptic weights that are related to the new hidden unit and the output bias weights only. However, in this study, we have not optimized the number of iterations that may he required to learn the input-output relationship for the particular problem.

6.2 Mathematical Model

We first define some variables as follows:

p is the" variable for the number of training patterns, where p = { 1,...P };

o is the variable for the output components with o={ 1,...0};

X. is the bias input which is kept fixed at 1;

x_j is the input signal with j={ 1...Ni);

$x_h(l)$ is the output from hidden unit l with l ={ Ni+1...Ni+n }. Here, Ni represents the input dimension, O the output dimension, and n is the number of" added hidden units (or the expanded input space). The energy function can, therefore, be written as:

$$E = \sum_{p=1}^{P} E^p = \sum_{p=1}^{P} \sum_{o=1}^{O} (t_o^p - y_o^p)^2 \qquad (1)$$

Let T be the target matrix, with a column for each input target pattern, given by:

$$T = \begin{bmatrix} t_1^1 \, t_1^2 \ldots t_1^P \\ \vdots \\ t_O^1 \, t_O^2 \ldots t_O^P \end{bmatrix}$$

and, the corresponding actual output matrix is:

$$Y = \begin{bmatrix} y_1^1 \, y_1^2 \ldots y_1^P \\ \vdots \\ y_O^1 \, y_O^2 \ldots y_O^P \end{bmatrix}$$

Then, with no hidden units in the network, one can calculate the output as:

$$Y = F(WX) \tag{2}$$

and let $\mathbf{W}_{io}$ be the set of weights between input and output matrices. *The best* estimation weight set of the given energy function (1) in affine space is calculated as:

$$W_{io} = F^{-1}(T)X^{+} \tag{3}$$

with X^{+} as the pseudo inverse of X[68], and F^{-1} as an inverse transformation matrix given by:

$$F^{-1}(T) = \begin{vmatrix} f^{-1}(t_1^1) \; f^{-1}(t_1^2) \; . \; . \; . \; f^{-1}(t_1^P) \\ \cdot \\ f^{-1}(t_O^1) \; f^{-1}(t_O^2) \; . \; . \; . \; f^{-1}(t_O^P) \end{vmatrix}$$

The set of weights $\mathbf{W}_{io}$ is then kept frozen. Assume that n hidden units arc added, and the output is calculated as follow:

$$y_o = f(net_o) \tag{4}$$

where,

$$net_o = \sum_{l=Ni+1}^{Ni+n} x_h(l) w_{lo} + \sum_{j=0}^{Ni} x_j w_j \ ;$$

and f is the transfer function of the output neuron (termed f_o). Further,

$$x_h(l) = f(net_h(l)) \tag{5}$$

where, f is the transfer function of the current hidden neuron (termed f_h and is equal to f_o),

$$net_h(l) = \sum_{k=Ni+1}^{Ni+l-1} x_h(k) w_{kl} + \sum_{j=0}^{Ni} x_j w_{jl}$$

and,

$$x_i = \left\{ \begin{array}{ll} 1 & if\ i = 0 \\ x_j & if\ i < Ni + 1 \\ x_h(l) & if\ i \geq Ni + 1 \end{array} \right\}$$

Let us define:

$$f_h'(l) = \frac{df(net_h(l))}{dnet_h(l)} \text{ ; and,}$$

$$f_o' = \frac{df(net_o)}{dnet_o}$$

With η as the learning rate, the stochastic gradient-descent gives the weight update as:

$$\Delta w_{ij} = -\eta \frac{\partial E^p}{\partial w_{ij}} \tag{6}$$

where, i and j denote the starting node i and the destination node j. And, applying the chain rule to Eqn. (6) for the weights between the hidden and the output neurons, and the bias synapses connected to the output, we get:

$$\frac{\partial E^p}{\partial w_{ij}} = \frac{\partial E^p}{\partial y_o^p} \frac{\partial y_o^p}{\partial net_o^p} \frac{\partial net_o^p}{\partial w_{ij}}$$

which can be written as, (we are only interested in the new hidden unit),

$$\frac{\partial E^p}{\partial w_{ij}} = -2(t_o^p - y_o^p) f_o'^p x_h^p(n) \tag{7}$$

Using Eqn (7), we can rewrite Eqn (6) explicitly with a first order and a second order term[69] as:

$$\Delta w_{no}(k) = \eta x_h^p(n)(t_o^p - o_o^p) f_o'^p - \alpha \Delta w_{no}(k-1) \tag{8}$$
$$\text{with } 0 < \alpha < 1$$

which gives the weight updates for the synaptic components between the currently added hidden unit n and the output o as shown in Figure 10. Similarly, the updates for the weights between the inputs (including expanded inputs and the bias weight at the currently added hidden unit) and the current hidden unit are given by:

$$\frac{\partial E^p}{\partial w_{ij}} = \frac{\partial net_h^p}{\partial w_{ij}} \frac{\partial f_h(net_h^p)}{\partial net_h^p} \sum_{o=0}^{O} \frac{\partial net_o^p}{\partial f_h(net_h^p)} \frac{\partial y_o^p}{\partial net_o^p} \frac{dE^p}{dy_o^p}$$

which then can be written as,

$$\frac{\partial E^p}{\partial w_{ij}} = -2 x_i^p f_h'^p(n) \sum_{o=1}^{O} w_{no} f_o'^p (t_o^p - y_o^p)$$

This equation is similarly written in an explicit form with a first order and a second order term as:

$$\Delta w_{in}(k) = \eta x_i^p f_h'^p(n) \sum_{o=1}^{m} w_{no}(t_o^p - y_o^p) f_o'^p + \alpha \Delta w_{in}(k-1) \tag{9}$$
$$\text{with } 0 < \alpha < 1$$

for the weight components. The change in learning rate after addition of each new hidden unit is given by:

$$\eta_{new} = \eta_{old} - \frac{c}{\#\ of\ iterations} \tag{10}$$

with η_{new} as the current learning rate, η_{old} as the previous learning rate, and c as a constant. When α is zero, we obtain the first order gradient descent and if α is a nonzero constant, then the two terms in both the Eqns (8) and (9) contribute to the weight updates and the second order gradient-descent is obtained.

6.3 Quantization of Weight Space

Because of the limited quantization weight space, the value ΔW_{ij} of the weight update will have to be modified to ΔW^*_{ij} to fit with the available quantization. The closeness between them will depend on the weight resolution available. Let *nbit be the* bit resolution of the weight space. Then the maximum level of weight space will be MAXLEVEL $=2^{(nbit)}-1$. We define *stepsize(n)* to be a step size for the weight space of a hidden unit *n* . The *stepsize(n)* can be generated from a constant *stepsize(0)* which is fixed before starting the learning process. The *stepsize(n) is* obtained as follows:

$$stepsize(n) = stepsize(0)\sqrt{\frac{E_{n-1}}{E_0}} \qquad (11)$$

with E_0 as the energy of the network without any hidden units or the bias input added (includes only the input to output weights calculated using pseudo-inverse technique), i.e.,

$$E_0 = \sum_{p=1}^{P} E'(W_{io};X^p;Y^p)$$

and the energy E_{n-1} is the energy of the network with n-1 hidden units added. There arc two ways to obtain the number of steps for ΔW_{ij}: one is the round-off technique where the number of steps are calculated as follows,

$$\#stepi = \left\{ \begin{array}{ll} (int)(\frac{\Delta W_{ij}}{stepsize(n)} + 0.5) & if\, \Delta W_{ij} > 0 \\ (int)(\frac{\Delta W_{ij}}{stepsize(n)} - 0.5) & if\, \Delta W_{ij} < 0 \end{array} \right\} \qquad (12)$$

and the other is the truncation technique for the calculation of the number of steps as below:

$$\#stepi = (int)(\frac{\Delta W_{ij}}{stepsize(n)}) \qquad (13)$$

Before updating the candidate weight using ΔW^*_{ij}, onc must ensure that the final quantized weight will not exceeded the limit provided as MAXLEVEL. Therefore, first the previously stored weight is converted to an equivalent number of steps, which is given by:

$$\#stepa = (int)(\frac{W_{ij}}{stepsize(n)}) \tag{14}$$

Then,

$$\Delta W^*_{ij} = \begin{cases} 0 & if\ (\#stepi + \#steps) > MAXLEVEL \\ stepsize(n)(\#stepi) & otherwise \end{cases} \tag{15}$$

6.4 Procedure for Learning in Hardware

A clear procedure for the learning algorithm, used for solution of a 6-bit parity problem, as an illustrative example, is presented below.

Based on the mathematical analysis of the EBP learning algorithm [69], the weight update (consisting of the first and second order terms) can be performed by incorporating (i) either the first order term only; or (ii) the summation of the two terms to obtain the second order effect as well. The idea of this development effort, of course, is to make the algorithm implementable in hardware given the limited synaptic weight resolution.

When considering the transfer characteristics of a neuron, a mathematical equivalent of sigmoid such as a logistic function is considered or a look-up table is constructed. A look-up table requites step updates and hence a quantization of the values. It has been shown that such a neuron quantization is not as sensitive as synaptic equalization [41,42] for the convergence properties of the circuit. In addition, the density of synapse on a chip is much higher than that of neurons. Thus, it is important to keep the synapse quantization as low as possible, commensurate with proper learning. Therefore, in our study, the effect of neuron quantization has not been considered. On the other hand, synaptic weight quantization is known to affect the sensitivity of learning to a larger extent, and the synaptic weights in hard ware maybe limited in their resolution any where from 5 to 10 bits.

6.5 Weight Update Issues:

The weight update Δw_{ij} is obtained as an analog number. However, the weight space is discrete in a hardware based on hybrid digital-analog synapse designs as is the case with our MDAC approach described earlier in Sec. 4.1. Therefore, to update the weight, the value Δw_{ij} must be converted into the number of steps by which the weight is to be updated before it is summed into the weight component. The conversion from an analog level to the respective discrete level results in some losses due to quantization.

In A/D conversion technique, typically, there are two conversion schemes: (1) Round-off; and (2) Truncation. In our simulation, we have compared these two schemes for their effectiveness in learning. We find that the round-off scheme works better in terms of convergence, especially when only the first order term in the weight update is considered. During learning phase, one must also consider the constraint of discrete levels limited by the available weight resolution. For example, with an 8-bit synapse, the number of discrete levels should not exceed 255.

Some of the salient features of our new learning algorithm arc:

1. The step size is dynamically changed after addition of each hidden unit. The change is based on the level of energy left over in the previous hidden unit (as a ratio of the original level of energy). In general, with the addition of a new hidden unit and subsequent training of the respective weights, the energy of the network decreases, resulting in smaller step size for the next stage of added hidden unit. However, in the present simulation for the 6-bit parity problem, the maximum number of hidden units added was limited to twenty irrespective of whether each additional hidden unit decreased the energy or not, or whether the network converged to the right solution.
2. The input to a neuron can be adjusted using two variables beside the input to the synapse itself. One is the weight value which can be updated during training, and the other is the bias itself to the synapse. It is this latter feature that allows for easy adjustment of the step size and, more importantly, promotes convergence with lower quantization of synaptic weights. Furthermore, this new design will provide independent, programmable, bias voltages to rows of synapses connected to each hidden unit.

7. THE 6-BIT PARITY PROBLEM

To assess the effectiveness of our methodology, and for easy comparison with other work reported in literature, we selected to study the 6-bit parity problem using our new CBP learning algorithm. The 6-bit parity problem has 64 discrete patterns to be classified. The neural network architecture has 6 inputs and one bias line directly connected to one output line through seven programmable weights. The procedure used for training is as follows:

1. Calculate the six weight values for the input-output connection weights using the pseudo-inverse relationship. In this particular case, the soI ution of the pseudo-inverse calculation is very close to zero. Therefore, we have arbitrarily set all the weights to 0.5. These weights are then kept frozen throughout.
2. Provide the input patterns (with bias weights not connected) and evaluate the respective output errors and calculate the energy E(0). If the errors are within a given tolerance, then the training is complete. If not, proceed further.
3. Set a learning rate, η =3.5 and α = 0.9 for second order effects and α = O for first order effects respectively, and a weight step size given by:
   stepsize(0) = 0.015*2@-~b@; where ***nbit*** = synaptic weight resolution in bits.
4. Add a new hidden unit along with randomly selected input and output weights, including the bias weights. These weights have to be converted to quantized levels of weights where each weight = ***stepsize(n)*** * *(#steps.)* Further, ***#stepa*** should be an integer given by either round-off or truncation method.
5. Again provide the inputs, measure the output, and evaluate the new error values for all the input patterns to ascertain if training is complete. Otherwise, continue the training process.
6. $\eta = \eta$ -3.5/10000, and ***stepsize(n)*** *is* given by equation (1 1).
7. Apply a random input pattern to the network.
8. Calculate the change in weights ΔW_{ij} using equations (8) and (9).
9. Calculate the number of steps required, **#stepi** using equations(12) and (1 3).
10. The total number of steps, ***#step(total)*** = *#stepa* + ***#stepi.*** If ***#step(total)*** > MAXLEVEL then set AW_{ij}* (= ***#stepi*** * ***stepsize(n)***) to 0. Otherwise, update W_{ij} and ***#stepa..***
    This procedure will update all the weights for the added hidden neurons and the output bias weights.
11. Go to 7, until the required application of number of iterations of the random patterns are completed. The number of iterations can be decided depending upon the requirement of the

problem and thc time available. In our case, we used 6000 iterations as an outer loop, and 64 iterations as an inner loop for each pattern.

12. Calculate the error for all the patterns and evaluate for completion of training. If complete, stop training, otherwise, calculate the energy E(n) and go to 4.

13 If the number of added hidden units is greater than 20, give up and quit.

7.1 Cascad e Backpropagation (CBP) Simulations

Using the above procedure, simulations for hardware were performed and the mean error and the standard deviation of the error were obtained for the four cases, two with only the first order term, with both round-off and truncation methods of conversion, and similarly the other two with the second order term included. As expected, the simulation showed that including the second order term made the errors go down considerably compared to that with just the first order term. As a result, this lcd to an acceptable solution with reduced synaptic weight resolutions. The mean error and the standard deviation curves for these four cases as an average of 10 runs are shown in Figures 10(a-d) and 11 (a-d) respectively. Overall, the method showed tremendous tolerance to reduced weight resolution and that with second order term included, the hardware with $\geq$ 7-bit resolution performed as good as that with full floating point accuracy with about 12 neurons added as hidden units. In addition, with the second order term included, the results with 6- and 7-bit resolution had closc to 100% correctness, and even 5-bit resolution weights provided 80 to 90% correctness. Table 2 summarizes these results of weight quantization and the correctness of the solution in the four cases (out of 64 patterns).

Table 2. Percent of correct CBP learning runs for the 6-bit parity problem with variation of synaptic weight resolution, using first order and second order terms in learning algorithm, with round-off (RO) and truncation (Tr) modes of weight value conversion)

	Percent correct, First order (RO)	Percent correct, First order (Tr)	Percent correct, Second o.(RO)	Percent correct, Second o(Tr)
5-bit weight Q	40%	10%	90%	80%
6-bit weight Q	90%	80%	100%	90%
7-bit weight Q	00%	80%	90%	1 00%
8-bit weight Q	00%	00%	1 00%	1 00%
9-bit weight Q	00%	00%	100%	1 00%
Floating point	00%	00%	100%	1 00%

8. HARDWARE IMPLEMENTATION WITH LEARNING

A neural network hardware system was assembled using the. analog neuron and synapse "building-block" chips described in Section 4.1, a computer interface, and CBP learning algorithm (Figure 12). The hardware consisted of eight building-block chips: two synapse-neuron hybrid chips gave a maximum of 64 neurons, two synapse-only chips completed this fully-parallel 64x64 architecture's first seven bits of synaptic precision. The last four synapse chips paralleled the first set of synapses in this network to allow larger d ynamic range in the weights (13 bits). A schematic diagram showing the chip arrangement is given in Figure 13. The system was connected to a personal computer, with parallel ports to access the digital weights, and analog converters to program inputs and read outputs. An early version of the CBP algorithm was used, with first-order learning dynamics and truncation of weights. The learning algorithm used all13 bits of precision (with the sign bits of each paralleled synapse pair tied together). Although the 13 bits thus obtained did not necessarily increase monotonically, the stochastic nature of the analog hardware evidently served to bridge nonmonotonicities sufficiently that learning could proceed. The measured input-output characteristics of one synapse-neuron pair is shown in Figure 14.

Two applications were tested, parity and a computation-intensive feature classification problem. The 2-bit parity problem was taught to the network by adding hidden units until the outputs were correct, exceeding a threshold of 3/5 full-range for a true and measuring below 2/5 for a false. This substantial noise margin at the output made it unlikely that noise levels would cause a false reading. After each hidden unit was added, 3000 backpropagation trials were executed, and network function was tested for correctness. A scatter plot of (binary) output as a function of (analog) inputs showed a marked bias towards "true" outputs overall, although the output was correct for the binary input representations (-Vmax, +Vmax) [24]. In most cases, 2-4 neurons were required as hidden units for all outputs to reach the criterion threshold levels.

The classification task, map separates, involves processing color map data (similar to roadmaps)- sampled at 24 bits per pixel- in order to determine the primary colors at each pixel. Representing maps with 24 bits per pixel is grossly wasteful, since a map is printed with only about eight colors, and thus, each pixel could be represented fully either by an 8-bit vector (if more than one color can be associated with a pixel) or a 3-bit color identifier (if each pixel is to be classified as only one color). Not only can storage requirements of maps can thus be reduced, but automatic and manual operations applied to the map would be greatly simplified if individual colors (such as black and red, mostly representing roads) could be independently extracted for display or processing. This task is more difficult than it may appear, since large variations in hue and intensity exist between maps, and even over one map. Furthermore, noise in the form of small dots may need to be filtered. Thus, a network must take as input a window of pixels centered about the particular pixel currently being processed. This application is well-suited to a fully-parallel neural networks due to the massive number of maps (currently stored on videodisk and CD-ROM memory) and the large number of pixels that need to be processed per map[70].

To map this problem into our hardware, a 3x3 window with 3 analog color intensities (i.e. red, green, and blue) pcr pixel was applied to the 27 inputs of the network, once for each pixel in our 305x200 map segment. The system was trained on a subset of the pixels that had been classified by hand, using a precursor of the CBP algorithm in which first-order truncation dynamics were used for weight mappings and error backpropagation, and all weights were adjusted at each step (rather than freezing all but the output layer). To validate the results, the error was compared with alternative classification methods, as shown in Table 3. Figure 15 shows the original map data and the neural net hardware output after completion of hardware-in-the-loop training. The hardware performed as well as the alternate classification algorithms running in software.

Table 3. A comparison of the accuracy of various data classification methods for the given map-data classification problem.

Classifier Method	Accuracy
Neural Network Simulation	91.2%
Neural Network Hardware	89.3%
K-nearest Neighbors	91.9%
Baysian-Unimodal Gaussian	89.8%

The primary limitation to processing speed in this application was the conversion between analog and digital domains at network input and output. Analog-to-digital conversion has frequently been the primary limiting factor in throughput, although one can expect that low-cost, high-speed video converters with sufficient dynamic range will soon become available.

9. CONCLUSIONS

Custom neural network hardware is appropriate, and in many cases required, for certain applications. When the application requires realtime response that exceeds the capability of high-end workstations, or when a mass-produced system that is not computer-based requires nontrivial data processing, custom neural hardware may be highly cost-effective. Furthermore, there are particular applications such as biological neural subsystem emulation [18], associative memory [71], and pattern matching [35] which beg a neural solution. But many general tasks can also be carried out by neural systems. Indeed, we look forward to the day when easy-to-use neural hardware "black boxes", each appropriate for certain classes and sizes of architecture, are available in much the same way that software objects capable of executing common (but perhaps specialized) tasks arc coming into usc today.

Both analog and digital implementations have their place in today's neural milieu. Digital designs allow a greater flexibility in mapping arbitrary or extremely large problems onto a limited number of time-multiplexed processing circuits, and no analog conversion is needed if inputs and outputs are already in binary representations. Also, advances in fabrication technologies may benefit digital implementations more so than analog. Analog circuits can be much smaller in size than digital, making it feasible to implement a fully parallel system of moderate size on one chip or a chipset. Taking advantage of physics, the circuits that aggregate signals at neuron inputs, which are space-consuming in digital technologies, can be implemented essentially as wires in analog technology. Also, while circuit noise may be considered by some to be the bane of analog implementations, many learning algorithms benefit from a noise component that can shake the weight configuration out of local minima. New techniques for obtaining larger silicon real-estate show promise for constructing very large networks. Wafer-scale neural networks have already been successfully constructed. It is only a matter of time until chip-stacking, which can be used to efficiently implement 3-D topologies, is used for neural networks.

Control of the network is usually relegated to a general purpose computer. This allows generality at a high level - the particular training set, learning algorithm, compensation for network flaws (such as malfunctioning output units), and the architecture that is mapped onto the resources available in the custom hardware can all be fine-tuned by the user. However, the greater the required learning and operation throughput, the costlier this processing overhead becomes (a phenomenon reminiscent of Amdahl's Law in computer engineering [72]). Thus, the highest throughput can only be obtained by directly interfacing the hardware to the external inputs and outputs, and implementing feedforward pass (and learning, if learning speed is critical) into a custom hardware, at the cost of generality. This unfortunate tradeoff of generality versus throughput may be somewhat ameliorated by implementing programmable datapaths and multiple learning algorithms in digital irnplementations (at a significant space penalty), or by using chipsets that allow a user wired-in flexibility over architecture in analog implementations.

The current state of custom neural hardware technology allows single-chip systems to be fabricated that have dozens of neurons and up to tens of thousands of synapses. A few specialized wafer-scale systems have exceeded these numbers by an order of magnitude. While such systems have been used to demonstrate a variety of interesting applications, only a handful of such networks have been commercialized for specific applications. Much application-development work has yet to be done before a niche for custom hardware can be carved out, It is likely that a good deal of this work must center on learning.

Learning algorithms are perhaps the "missing link" in the development of custom hardware implementations. While much effort has gone into the analysis and development of learning algorithms for computer-based neural simulations, relatively little work bass been directed toward developing or adapting learning algorithms to be compatible with the limited precision (and analog-system noise) inherent in hardware. Almost certainly, more work is needed in this direction before custom neural hardware can become mainstream. It can be anticipated that the learning algorithms used by biological systems will soon be more fully teased out, and that these algorithms will be profitably applied to hardware, especially to analog circuits.

Most learning algorithms in use today require a user to select the network architecture before learning commences. Looking towards the goal of semi-autonomous "black-box" networks, algorithms that automatically configure the network until a criterion level of performance is reached would be highly advantageous for hardware implementations. Two feed-forward learning algorithms that add hidden-unit layers automatically are CC and our CBP method. We developed CBP to simplify the hardware required for on-chip learning, and to allow learning with as few bits of synaptic precision as possible. We have shown that CBP can work reliably with as few as five bits of weight precision. To achieve this, a method for associating dynamic weight-update steps was developed that may be applied to any cascade learning algorithm as long as each layer can learn independently of the others.

Acknowledgments

The research described herein was performed by the Center for Space Microelectronics Technology, Jet Propulsion Laboratory, California Institute of Technology and was jointly sponsored by the Ballistic Missile Defense Organization/Innovative Science and Technology Office

(BMDO/IST), the Office of Naval Research (ONR), the Advanced Research Projects Agency (ARPA), and the National Aeronautics and Space Administration (NASA). The authors thank Drs. A. Stubberud, S. Kemeny, and R. Tawel, and S. Gulati for useful discussions. Authors also acknowledge M. Tran and H. Langenbacher for technical assistance.

References!

[1] M. Holler, S. Tam, H. Castro, and R. Benson, "An electrically Trainable Artificial Neural network (ETANN) with 10240 "floating gate" synapses," *Proc. IEEE IJCNN*, vol. II, June 18-22, 1989, Washington, DC, pp. 191-196.

[2] Y. Arima, et al. "A Refreshable Analog VLSI Neural Network Chip with 400 Neurons and 40K Synapses," *IEEE Journal of Solid-State Circuits,* Vol. 27, NO. 12, pp. 1868-1875, Dee, 1992.

[3] D. Hammerstrom, "A Massively Parallel Architecture for Cost-Effective Neural Network Pattern Recognition, Image processing, and Signal processing Applications". *Digest of Papers in GOMAC Conference,* pp. 281-284, 1992, Las Vegas, Nevada.

[4] C. Grove "Ni 1000 Recognition Accelerator chip™" Netstor, Inc. One Rchmond Square, Providence, RI 02906.

[5] N.H. Farhat, "Optoelectronic neural networks and learning machines," *IEEE Circuits and Devices Msg.*, Sept. 1989, pp 32-41.

[6] T. Daud, A. Moopenn, J. Lamb, A. Thakoor, and S. Khanna, "Feedforward, high density, programmable read only neural network breed memory system," SPIE/ *High Speed Computing,* Ed: D.P. Cassasent, Vol. 880, 11-12 Jan., 1988, Los Angeles, CA, pp 76-84.

[7] A.P. Thakoor, A. Moopenn, J. Lambe, and S.K. Khanna, "Electronic hardware implementations of neural networks," *Applied Optics,* Vol. 26,5085-5092, 1987.

[8] R. Ramesham, T. Daud, A. Moopenn, A.P. Thakoor, and S.K. Khanna, "Manganese oxide microswitch for electronic memory based on neural networks," *J. Vat. Sci. Technol.* B 7 (3), 450-454, 1989.

[9] S. Thakoor, A. Moopenn, T. Daud, and A.P. Thakoor, "Solid state thin film memistor for electronic neural networks," *J. Appl. Phys. 67, 3132,* 1990.

[10] A. J. Agranat, C. F. Neugebauer, R. B. Nelson, and A. Yariv, "The CCD Neural Processor: A Neural Network Integrated Circuit with 65536 Programmable Analog synapses," *IEEE Trans. Circuits Sys.* Vol. 37, NO. 8,pp. 1073-1075, Aug. 1990.

[11] A. M. Chiang and J. R. LaFranchise "Real-time CCD-Based Neural Network Systems for Pattern Recognition Applications," *Digest of Papers in GOMAC Conference,* pp. 285-288, 1992, Las Vegas, Nevada.

[12] A.F. Murray, "Pulse arithmetic in VLSI neural networks," *Micro Magazine,* p 64-74, Dec. 1989.

[13] J. E. Tomberg and K.K.K. Kaski, "Pulse-density modulation technique in VLSI implementations of neural network algorithms," *IEEE J. of Solid-State Circuits,* vol 25, no. 5, Ott 1990, pp 1277-1286.

[14] S. Churcher, et al. "Generic Analog Neural Computation-The EPSILON Chip," In: *Advances in Neural Information Processing Systems,* Vol. 5, pp. 773-780, Morgan Kaufman, San Mateo, CA, 1993.

[15] R. Sarpeshkar, et al. "Visual Motion Computation in Analog VLSI using Pulses," In: *Advances in Neural Information Processing Systems,* Vol. 5, pp. 781-788, Morgan Kaufman, San Mateo, CA,1993.

[16] Hamilton, A., A.F. Murray, D.J. Baxter, S. Churcher, H.M. Reckie and L. Tarassenko, "Integrated Pulse Stream Neural Networks: Results, Issues and Pointers," *IEEE Trans. Neural Networks, 3(3); 385-393 (1992).*

[17] Hopfield, J.J, "The effectiveness of analogue 'neural network' hardware," *Network 1*; 27-40 (1990) (IOP Publ. Ltd., U.K.).

[18] C. Mead, <u>*Analog VLSI and Neural Systems*</u>, Addison-Wesley, Reading, MA, 1989.

[19] S. Schoenung, B. S. Papadales, and T. A. Tibbetts, Data compiled by W.J. Schafer Associates in conjunction with a study jointly funded by JPL and BMDO.

[20] Yasunaga, M., N. Masuda, M. Yagyu, M. Asai, K. Shibata, M. Ooyama, M. Yamada, T. Sakaguchi and M. Hashimoto, "A Self-Learning Digital Neural Network Using Wafer-Scale LSI," *IEEE J. Solid-State Circuits 28(2); 106-114* (1993).

[21] Masaki, M., Y. Hirai and M. Yarnada, "Neural Networks in CMOS: a Case Study," *IEEE Circuits and Devices Mug. 6(4);* 13-17 (1990).

[22] R.C. Frye, E.A. Rietman, and C.C. Wong, "Back-propagation learningand nonidealities in analog neural network hardware," *IEEE Trans. on Neural Networks,* Vol 2; NO. 1, pp 110-117, Jan, 1991.

[23] Carley, L. R., "Trimming Analog Circuits Using Floating-Gate Analog MOS Memory," *IEEE J. Solid-State Circuits 24(6);* 1569-1575 (1989).

[24] T. A. Duong, S. P. Eberhardt, M. D. Tran, , T. Daud, and A. P. Thakoor, "Learning and Optimization with Cascaded VLSI Neural network Building-Block Chips," *Proc. IEEE/INNS International Join Conference on Neural Networks,* June 7-11,1992, Baltimore, MD, vol. I, pp. 184-189.

[25] S.P. Eberhardt, T.A. Duong, and A.P. Thakoor, "Design of parallel hardware neural network systems from custom analog VLSI "building-block" chips," *IEEE/INNS Proc. IJCNN,* June 18-22, 1989 Washington D. C., vol. II, pp. 183.

[26] A.P. Moopenn, T.A. Duong, and A.P. Thakoor, "Digital-Analog Hybrid Synapse Chips for Electronic Neural Nteworks," *Adv. in Inf. Proc. Sys. 2,* Ed: D.S. Touretzky, Morgan Kaufmann, 1990, pp. 769-776.

[27] F. Kub, L Mack, K. Moon, C. Yao, and J. Modolo, "Programmable analog synapses for microelectronic neural networks using a hybrid digital-analog approach", poster presented at *IEEE Int'l Conf. Neural Networks*, San Diego, July 24-27, 1988.

[28] M. Jabri and B. Flower "Weight Perturbation: An Optimal Architecture and learning Technique for Analog VLSI Feedforward and recurrent Multilayer Networks," *IEEE Trans. on Neural Networks,* Vol 3, NO. 1, pp 154-157, Jan, 1992.

[29] T.A. Duong, "On-chip learning in VLSI hardware", *NASA Technology Briefs,* (to be published).

[30] S.P. Ebcrhardt, T.A. Duong, and A.P. Thakoor, "A VLSI synapse "building-block" chip for hardware neural network implementations," *Proc. Third Annual Parallel Processing Symposium,* Mar. 1989, Fullerton, CA, vol. I, pp. 257-267, IEEE Orange County Computer Society.

[31] Benson, R.G. and D.A. Kerns, "UV-Activated Conductance Allow For Multiple Time Scale Learning," *IEEE Trans. Neural Networks 4(3); 434-440 (1993).*

[32] Madani, K., P. Garda, E. Belhaire and F. Devos, "Two Analog Counters for Neural Network Implementation," *IEEE J. So/id-State Circuits 26(7); 966-974 (1991).*

[33] Ghosh J., P. Lacour and S. Jackson, "OTA-based Neural Network Architectures with On-Chip Tuning of Synapses," *IEEE Trans. Circuits and Syst.* - II Analog and Digital Sign. Proc. 41(1); 50-57 (1994).

[34] Lee, B.W. and B.J. Sheu, "General-Purpose Neural Chips with Electrically Programmable Synapses and Gain-Adjustable neurons," *IEEE J. Solid-State Circuits 27(9);* 1299-1302(1992).

[35] B.E. Boser, E. Sackinger, J. Bromley, Y. LeCun, and L.D. Jackel, "An Analog Neural Network Processor with Programmable Topology," *IEEE Journal of Solid State Circuits,* vol. *26, NO. 12,* Dec. 1991.

[36] 80170NW Electrically Trainable Neural Network Specification Sheet, USA/E358/0590/2k/ GF/CC, Intel Corp. (1990).

[37] S.P. Eberhardt, T.A. Duong, R. Tawel, F.J. Pineda, and A.P. Thakoor, "A robotic inverse kinematics problem implemented on neural network hardware with gradient-descent learning," *Proc. of the 2nd. ISTED International Symposium on Expert Systems and neural Networks,* Hawaii, Aug. 15-17, 1990, Ed: M.H. Hamza, pp. 70-73

[38] T. A. Duong, T. X. Brown, M. D. Tran, S. P. Eberhardt, T. Daud, and A. P. Thakoor, "Cascaded VLSI Neural network Building-Block chips for Map Classification," *Digest of Papers,*

Goverment Microcircuit Applications Conference, Las Vegas, NV, Nov. 10-12, 1992, pp. 145-146.

[38] T. A. Duong et. al,"Low Power Analog Neurosynapse Chips for a 3-D "Sugarcube" Neuroprocessor," *Proc. of IEEE Intl' Conf. on Neural Networks*(ICNN/WCCI), Vol 111, pp. 1907-1911, June 28-July 2, 1994, Orlando, Florida

[40] Shanken, S. N., "3-D processor packaging and interconnect," *Proceedings of the Government Microcircuits Applications Conference,* Vol. XVII, pp 151-154, 1991.

[41] M. Hochfeld and S. Fahlman, "Learning, with limited numerical precision using the cascade-correlation algorithm ," *IEEE Trans. Neural Networks,* vol.3, *No. 4,* pp 602-611, July 1992.

[42] P. W. Hollis, J.S. Harper, and J.J. Paulos, "The effects of Precision Constraints in a Backpropagation learning Network," *Neural Computation, vol. 2,* pp. 363-373, 1990.

[43] S. Sakaue, T. Kohda, H. Yamamoto, S. Maruno, and Y. Shimeki, "Reduction of required precision bits for back propagation applied to pattern recognition," *IEEE Trans. Neural Networks,* Vol. 4, No. 2, pp270-275, March 1993.

[44] R. Tawel, "Learning in analog neural network hardware," *Computers Elect. Engng,* Vol. 19, No. 6, pp 453-467, 1993.

[45] S. P. Eberhardt, R. Tawel, T. X. Brown, T. Daud, and A. P. Thakoor, "Analog VLSI Neural Networks: Implementation Issues and Examples in Optimization and Supervised Learning," *IEEE Trans. Indust. Electron.* vol. 39, no. 6,pp. 552-564, Dec. 1992.

[46] Andes, D., B. Widrow, M. Lehr and E. Wan, "MRIII: A Robust Algorithm for Training Analog Neural Networks," *Proc. Int'l Joint Neural Networks Conf.,* Vol. 1, Washington, D.C, Jan 15-19; 533-536 (1990).

[47] T. Duong, T. Daud, and A. Thakoor, " Cascaded VLSI Neural Network Architecture for On-line Learning," *NASA Tech Brief* , vol. 17, NO 12, Dee, 1993.

[48] B. Furman and A. Abidi, “A CMOS backward error propagation LSI," *Proc. 22nd Asilomar Conf. on Signal, Systems, and Computers,* Pacific Grove, CA, Nov, 1988.

[49] Eberhardt, S.P., Analog Hardware for Delta-Backpropagation Neural Networks,” U.S. Patent 5,101,361 (1992).

[50] Moric, T. and Y. Amemiya, “An All-Analog Expandable Neural Network LSI with On-Chip Backpropagation Learning,” *IEEE J. Solid-State Circuits 29(9); 1086-1093 (1 994).*

[51] Lent, J.B. and W. Guggenbuehl, “Analog CMOS Implementation of a Multilayer Perception with Nonlinear Synapses,” *IEEE Trans. Neural Networks 3(3); 457-465 (1992).*

[52] Shims, T., T. Kimura, Y. Kamatani, T. Itakura, Y. Fujita and T. Iida, "Neuro Chips with On-Chip Back-Propagation and/or Hebbian Learning,” *IEEE J. So/id-State Circuits, 27(12);* 1868-1876 (1992).

[53] D. B. Kirk, et al. “Analog VLSI Implementation of Multi-dimensional Gradient Descent,” In *Advances in Neural Information Processing Systems,* Vol. 5, pp. 789-796, Morgan Kaufman, San Mateo, CA.

[54] Donald, J. and L. Akers, “An Adaptive Neural Processing Node,” *IEEE Trans. Neural Networks 4(3); 413-426 (1993).*

[55] He, Y. and U. Cilingiroglu, “A Charge-Based On-Chip Adaptation Kohonen Neural Network,” *IEEE Trans. Neural Networks 4(3); 462-469 (1993).*

[56] Heim, P., and E. A. Vittoz, “Precise Analog Synapse for Kohonen Feature Maps,” *IEEE J. Solid-State Circuits 29(8)* ;982-985 *(1994).*

[57] Macq, D., M. Verleysen, P. Jespers and J.-D. Legat, “Analog Implementation of a Kohonen Map with On-Chip Learning,” *IEEE Trans. Neural Networks* 4(3); 456-461 (1993).

[58] Widrow, B., “Generalization and Information Storage in Networks of Adeline ‘Neurons’,” in *Self-Organizing Systems, M.* Yovitz, G. Jacobi and G. Goldstein, eds. Washington D. C.: Spartan Books, pp. 435-461 (1962).

[59] J. Alspector, J.W. Gannett, S. Haber, M.B. Parker, and R. Chu, "A VLSI-efficient technique for generating multiple uncorrelated noise sources and its application to stochastic neural networks," *IEEE Trans. on Circuits and systems, vol. 38, NO.* 1, pp. 109- 123,Jan, 1991.

[60] J. R. Alspector, et al. "A parallel Gradient Descent Method for Learning in analog VLSI Neural Networks," In *Advances in Neural Information Processing Systems*, Vol. 5, pp. 836-844, Morgan Kaufman, San Mateo, CA.

[61] A. Jayakumar and J. Alspector, "On-Chip Learning in Analog VLSI Using Simulated Annealing," *Digest of Papers* in *GOMAC Conference,* pp. 277-280, 1992, Las Vegas, Nevada.

[62][Fang, W.-C., B.J. Sheu, O. T.-C. Chen and J. Choi, "A VLSI Neural Processor for Image Data Compression Using Self-Organizing Networks," *IEEE Trans. Neural Networks 3(3); 506-518 (1992).*

[63] Kirkpatrick, C.G., R.C Kezer and G.A. Works, "Intelligent Gradient Descent IC with On-Chip Learning," *Digest of Papers in GOMAC Conference*, pp. 273-276, 1992, Las Vegas, Nevada.

[64] Uchimura, K., O. Saito and Y. Amemiya, "A High-Speed Digital Neural Network Chip with Low-Power Chain-Reaction Architecture," *IEEE Trans. Solid-State Circuits 27(1 2);* 1862-1867 (1 992).

[65] S. E. Fahlmann, C. Lebiere, "The Cascade Correlation learning architecture," in *Advances in Neural Information Processing Systems II*, Ed: D. Touretzky, Morgan Kaufmann, San Mateo, CA, 1990, pp. 524-532.

[66] Chen, A. M., and Hecht-Neilsen, R "On the geometry of feedfordward neural network of weight spaces." In *Proceeding of the 2nd of IEE Conference on Artificial Neural Networks,* pp. 1-4. IEE Press, London.

[67] T.A. Duong, "An Analysis of cascade architecture in neural network learning," Under preparation.

[68] G. Strang, *Linear Algebra and its applications* , 3rd *edition,* Harcourt *Brace* Jovanovich, Publishers, San Diego, 1988.

[69] D.B. Parker, "Optimal Algorithms for Adaptive Networks: Second order Back Propagation, Sececond order Direct Propagation, and Second Order Hebbian Learning," *Proc. IEEE First Intel' Conf. Neural Networks,* Vol. II, pp. 593-600, San Diego, CA, 1987.

[70] T.A. Duong, T. Brown, M. Tran, H. Langenbacher, and T. Daud, "Analog VLSI neural network building block chips for hardware-in-the-loop learning," *Proc. IEEE/INNS Int'l Join Conf. on Neural Networks,* Bejing, China, Nov. 3-6, 1992.

[71] Verleysen, M., B. Sirletti, A. M. Vandemeulebroecke and P. G. A. Jespers, "Neural Networks for High-Storage Content-Addressable Memory: VLSI Circuit and Learning Algorithm," *IEEE J. Solid-State Circuits 24(3); 562-569, 1989.*

[72] Wawrzinek, J., K. Asanovic and N. Morgan, "The design of a Neuro-Microprocessor," *IEEE Trans. Neural Networks 4(3); 394-399 (1993).*

--------------------------------- -----------

Figure 1 A schematic block diagram showing the synapse (square blocks) and neuron (triangles) functions and signal flow. For maximum generality, outputs could be connected on-chip to inputs, and a feedforward network could be mapped onto this architecture by nulling synapses leading from any layer to the neurons in the same or earlier layer(s).

Figure 2 Block diagram of a capacitor refresh 32 x 32 synapse chip showing arrangement of individual synapse cells with signal flow, and row and column decoders.

Figure 3 A circuit diagram of a 7-bit digital-analog hybrid synapse cell using scaled current morrors to implement a monotonic programmable scaling circuit. The in.set shows schematically the digital circuit for weight storage.

Figure4. Circuit diagram of a wide range variable gai n neuron with sigmoidal transfer characteristics. The stages of voltage to current conversion, comparator, and gain controllers are shown. Input potential is kept constant to avoid capacitive charge delay, and the gain control stage allows programming sigmoidal slope.

Figure 5 Conceptual diagram of a 3-dimensionally stacked multichip module integrated with a 2-D focal plane array. All the sensor array contacts are bump-bonded to the chip-stack under it.

Interconnections (including controls and power) at the neural net chips are brought out along the edges to the connecting busbars.

Figure 6 Circuit diagram for a high speed and compact variable gain neuron cell. A feedforward time of < 150 nanosecond per synapse-neuron pass is more than an order of magnitude improvement over previous design with a 6-7 microseconds delay.

Figure 7 Synapse output current as a function of synaptic weights (±63 levels) with different input voltages, V_{in} (1 .5, 1.9, 2.4, and 2.8 volts).

Figure 8 Synapse-neuron sigmoidal transfer curves giving neuron output voltage as a function of synaptic weights (±63 levels) with neuron gain as a parameter.

Figure 9 A schematic diagram of a cascade backprop architecture showing added hidden units. Synaptic weights arc shown as small rectangles where the filled rectangles signify that they are frozen after completion of training, before the next hidden unit is added.

Figure 10 Mean error as a function of added hidden units with synaptic weight resolution as a parameter for the cascade backpropagation (CBP) learning simulation for hardware with (a) only the first order weight update term and round-off (fob-) conversion; (b) first order term and truncation (fo/t) conversion; (c) including second order term and round-off (so/r) conversion; and (d) second order term and truncation (so/t) conversion methods.

Figure 11 Standard deviation as a function of added hidden units with synaptic weight resolution as a parameter for the cascade backpropagation (CBP) learning simulation for hardware with (a) only the first order weight update term and round-off (fo/r) conversion; (b) first order term and truncation (fo/t) conversion; (c) including second order term and round-off (so/r) conversion; and (d) second order term and truncation (so/t) conversion methods.

Figure 12 A photograph of an eight chip board with 64x64, 13-bit synaptic array and 64 neurons connected to a host computer and configured as a feedforward net for solution of a map-data classification problem with hardware-in-the-loop cascade backprop learning.

Figure 13 A schematic diagram showing 2 neuron-synapse and 6 synapse chips cascaded to form a 64x64 reconfigurable neural network with nominally 13 bits of synaptic resolution.

Figure 14 Synapse-neuron sigmoidal transfer curves for the cascaded high resolution synapse arrangement with ±4096 levels

Figure 15 Map-data input (left) and feature classified output (right) for a 305x200-pixel map-segment using the setup shown in Fig. 13. A training set of 2000 data-points was used, and the neural network architecture consisted of 27 analog inputs for each 3x3-pixel window, each pixel with 3 colors (RGB), each color with an 8-bit of resolution (256 levels). The output consisted of 7 different color outputs (< 3 bits) assigned to each central pixel of the 3x3 window representing as many different features on the map. The hardware solution with 89.3% accuracy was nearly as good as that obtained using other feature classification methods in software (Table 3).

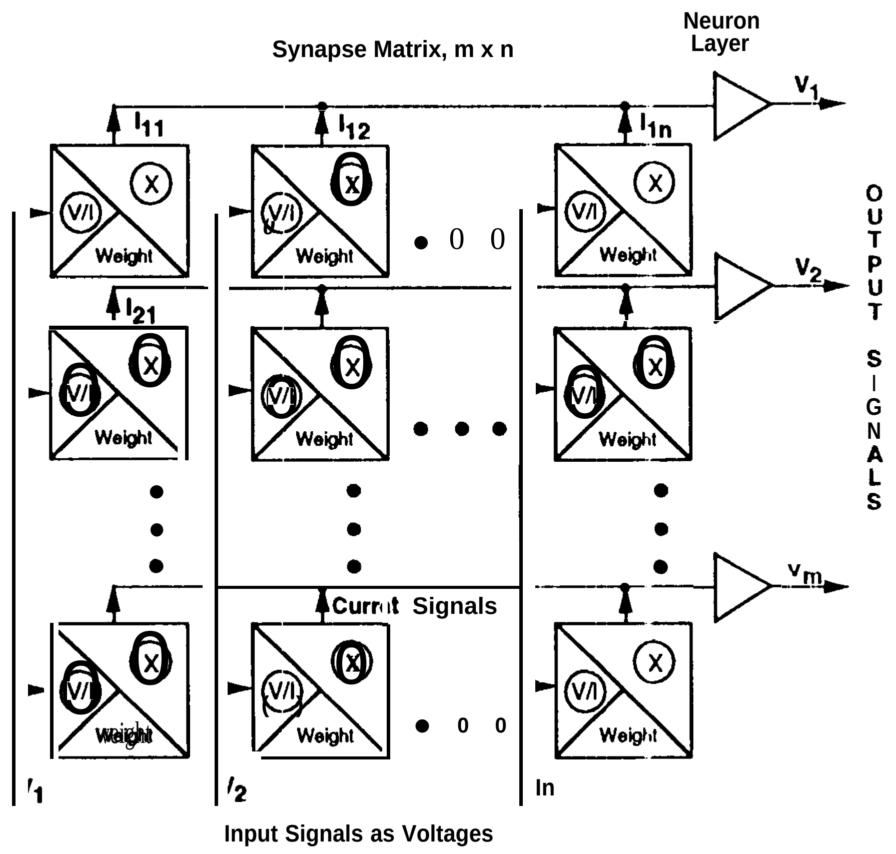

Neuron Layer
Synapse Matrix, m x n
I_{11}
I_{12}
I_{1n}
I_{21}
V/I
X
Weight
V_1
V_2
V_m
OUTPUT SIGNALS
Current Signals
I_1
I_2
In
Input Signals as Voltages

Figure 1

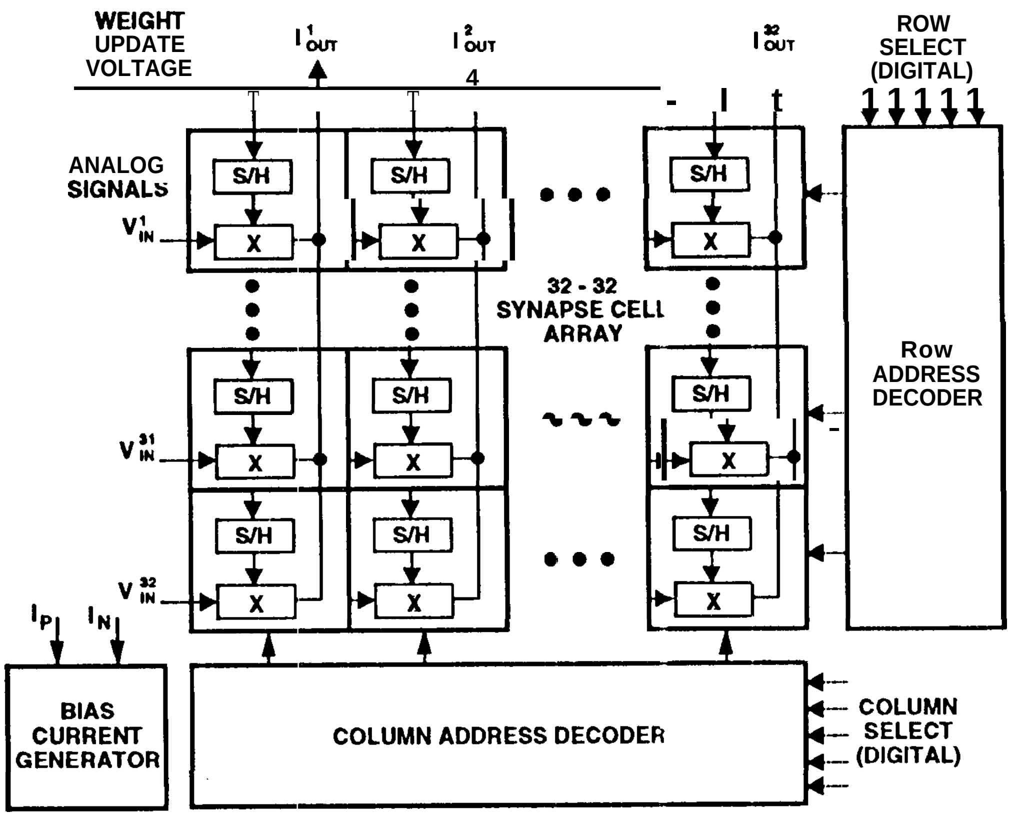
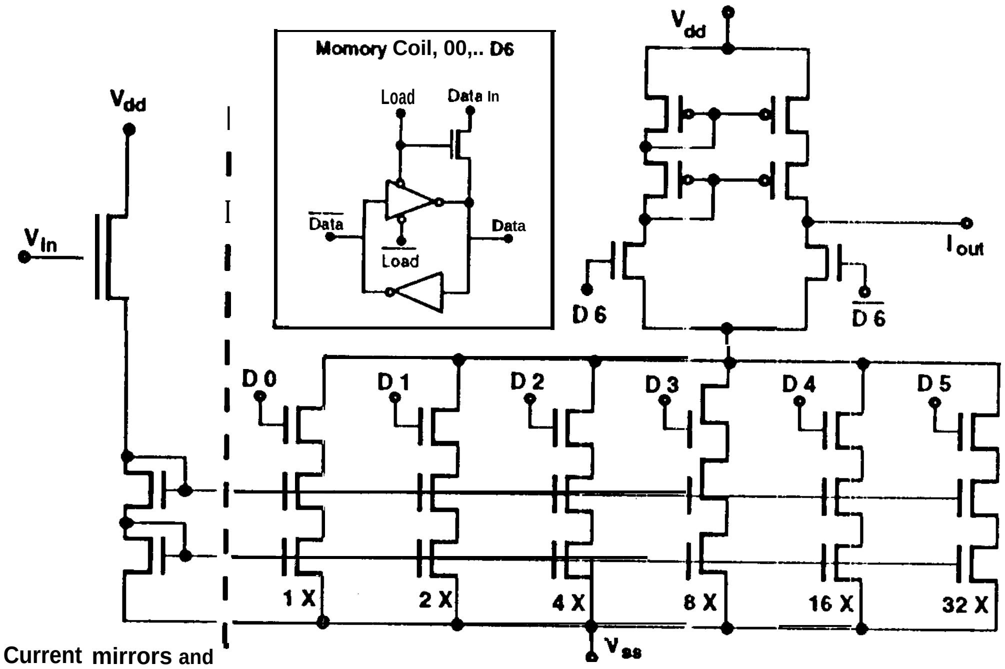

WEIGHT UPDATE VOLTAGE
I^1_{OUT}
I^2_{OUT}
I^{32}_{OUT}
ROW SELECT (DIGITAL)
ANALOG SIGNALS
V^1_{IN}
V^{31}_{IN}
V^{32}_{IN}
S/H
X
32 - 32 SYNAPSE CELL ARRAY
Row ADDRESS DECODER
I_P
I_N
BIAS CURRENT GENERATOR
COLUMN ADDRESS DECODER
COLUMN SELECT (DIGITAL)

Figure 2



Current mirrors and V/I common to all the **synapses** along a row In the matrix

Individual synapse circuits with digital weight storage through ON/OFF **latches** Do - D5, and **+/- current** direction through latch .6 giving a 7-bit (6 + sign bit) resolution

Figure 3

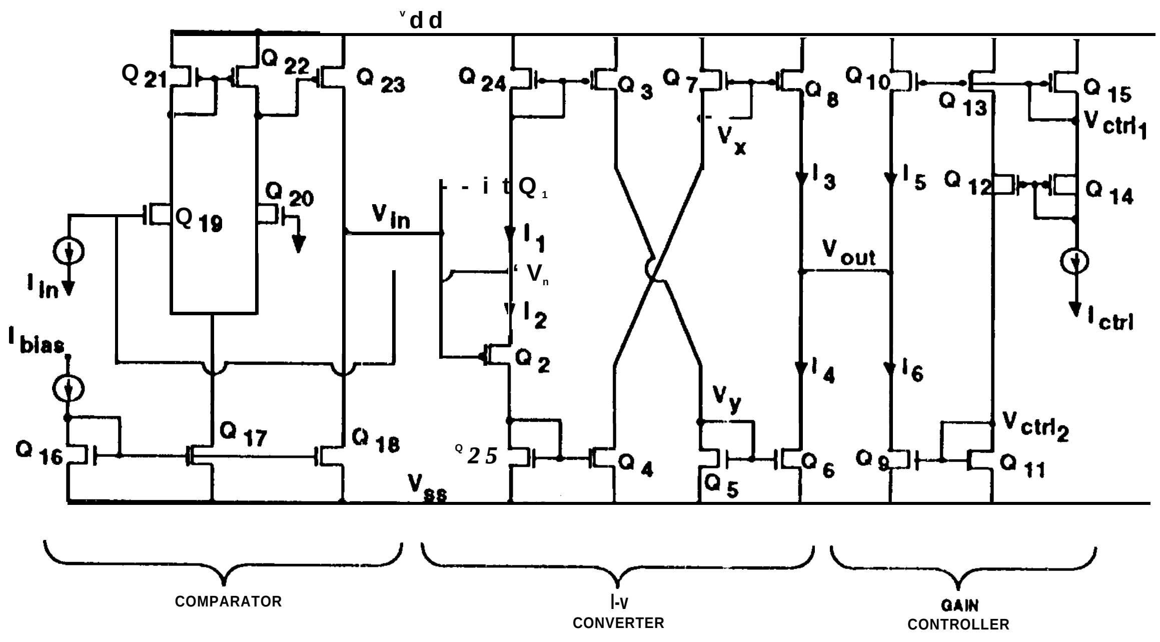

Vdd
Q 21
Q 22
Q 23
Q 24
Q 3
Q 7
Q 8
Q 10
Q 13
Q 15
V ctrl1
V x
I 3
I 5
Q 12
Q 14
Q 19
Q 20
V in
Q 1
I 1
V n
I 2
Q 2
V out
I in
I bias
I 4
I 6
I ctrl
V y
V ctrl2
Q 16
Q 17
Q 18
Q 25
Q 4
Q 5
Q 6
Q 9
Q 11
V ss
COMPARATOR
I-V
CONVERTER
GAIN
CONTROLLER

Figure 4

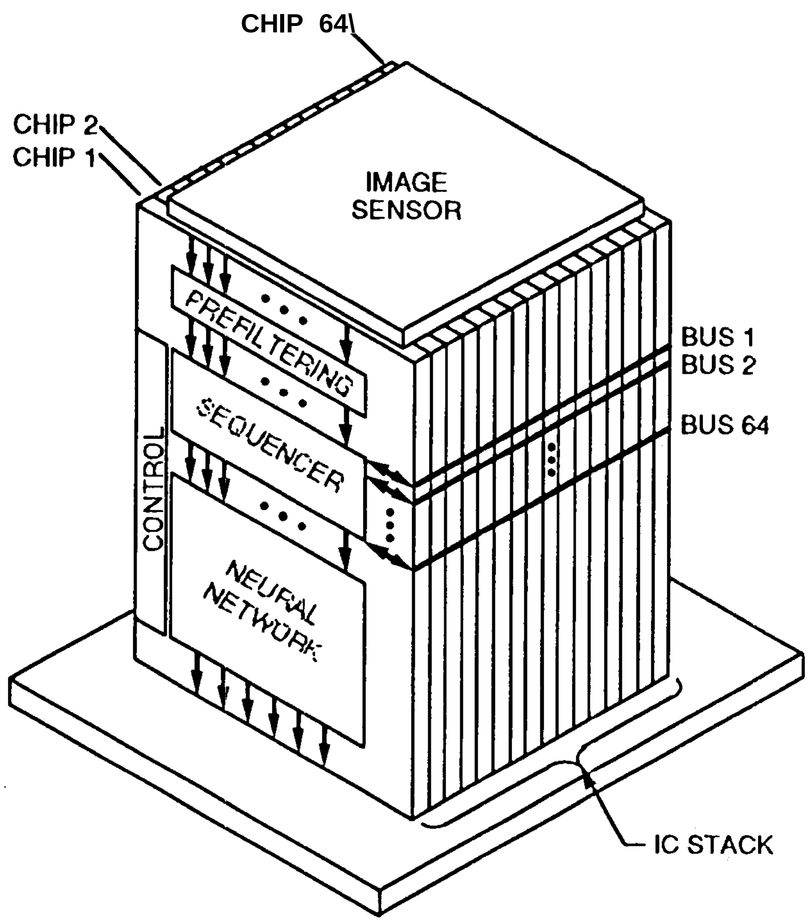

CHIP 64
CHIP 2
CHIP 1
IMAGE SENSOR
PREFILTERING
SEQUENCER
CONTROL
NEURAL NETWORK
BUS 1
BUS 2
BUS 64
IC STACK

Figure 5

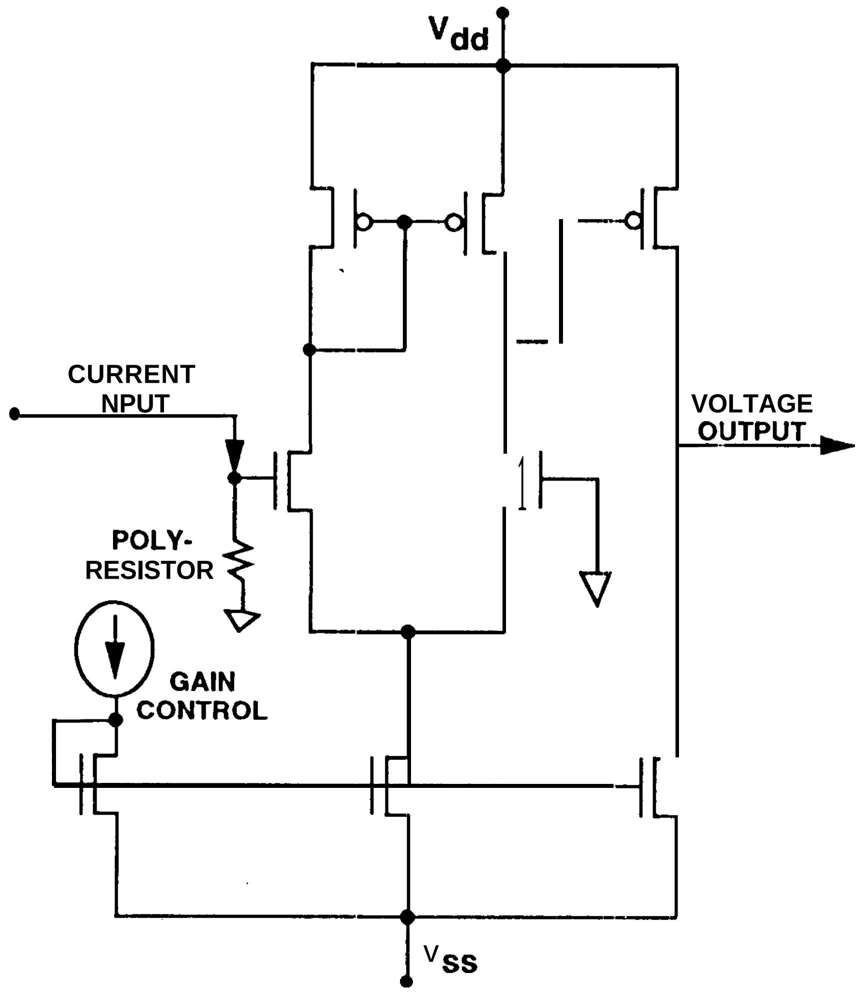

V dd
CURRENT
NPUT
VOLTAGE
OUTPUT
POLY-
RESISTOR
GAIN
CONTROL
V SS

Figure 6

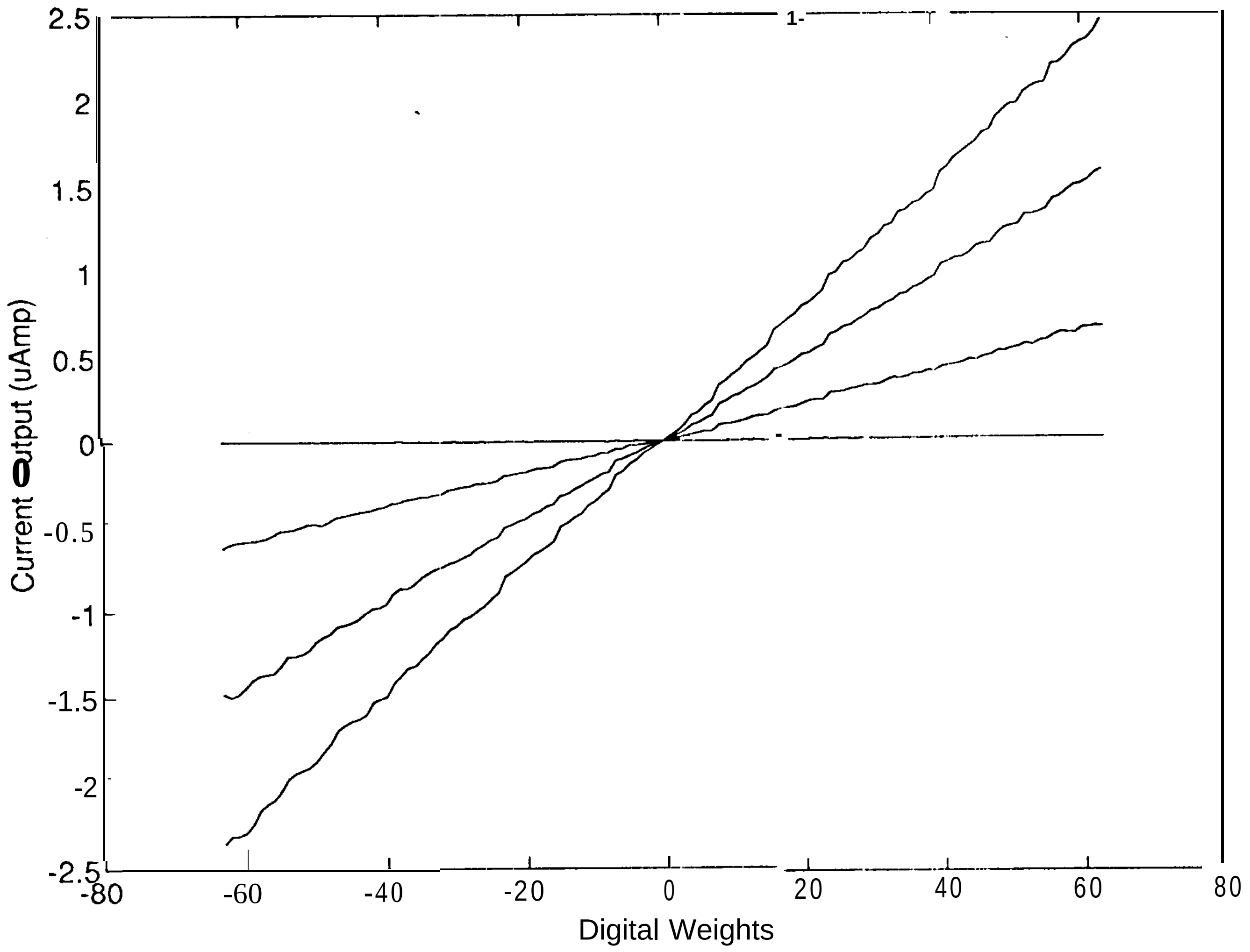

2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
Current Output (uAmp)
-80
-60
-40
-20
0
20
40
60
80
Digital Weights

Figure 7

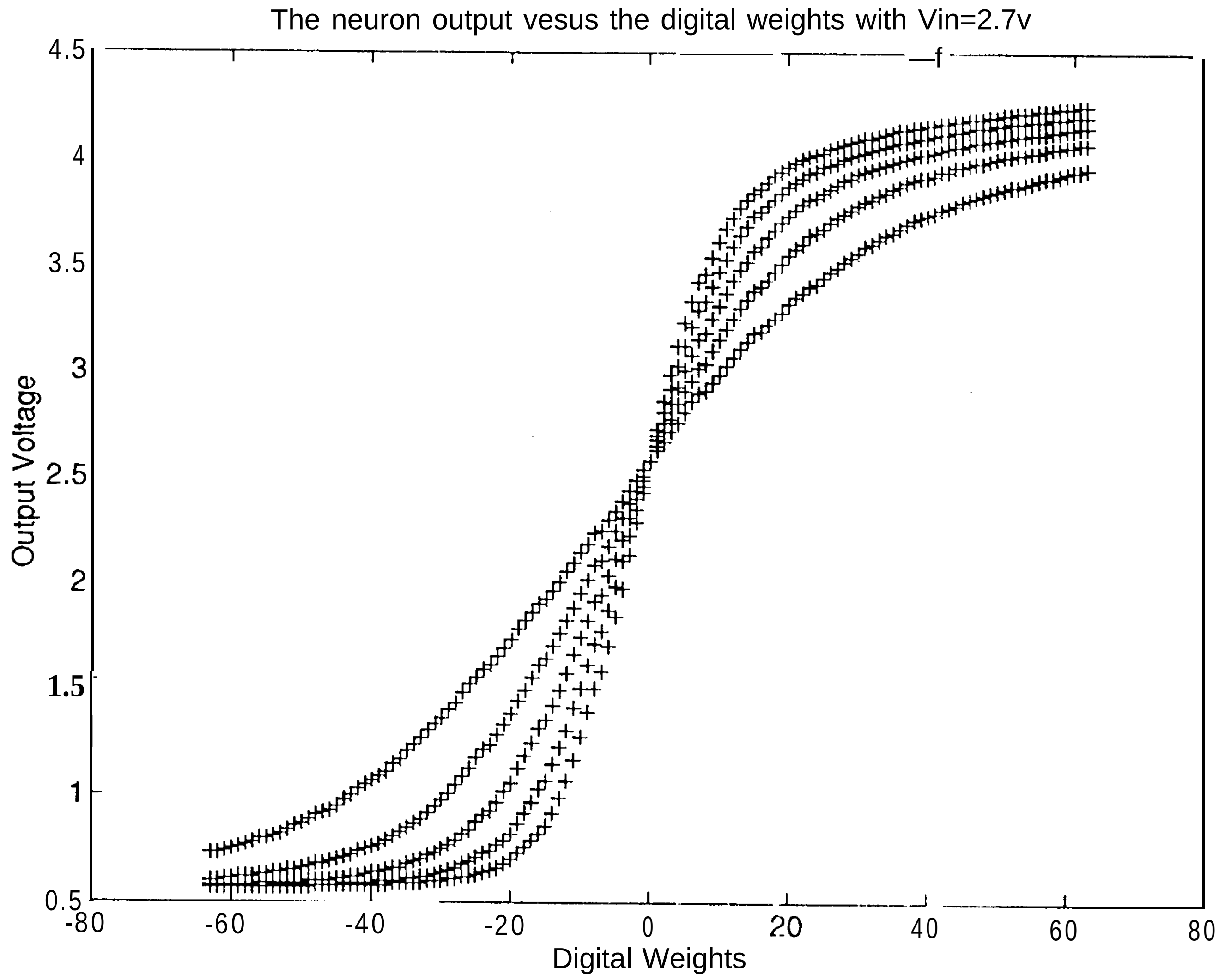

The neuron output vesus the digital weights with Vin=2.7v
Output Voltage
4.5
4
3.5
3
2.5
2
1.5
1
0.5
-80
-60
-40
-20
0
20
40
60
80
Digital Weights

Figure 8

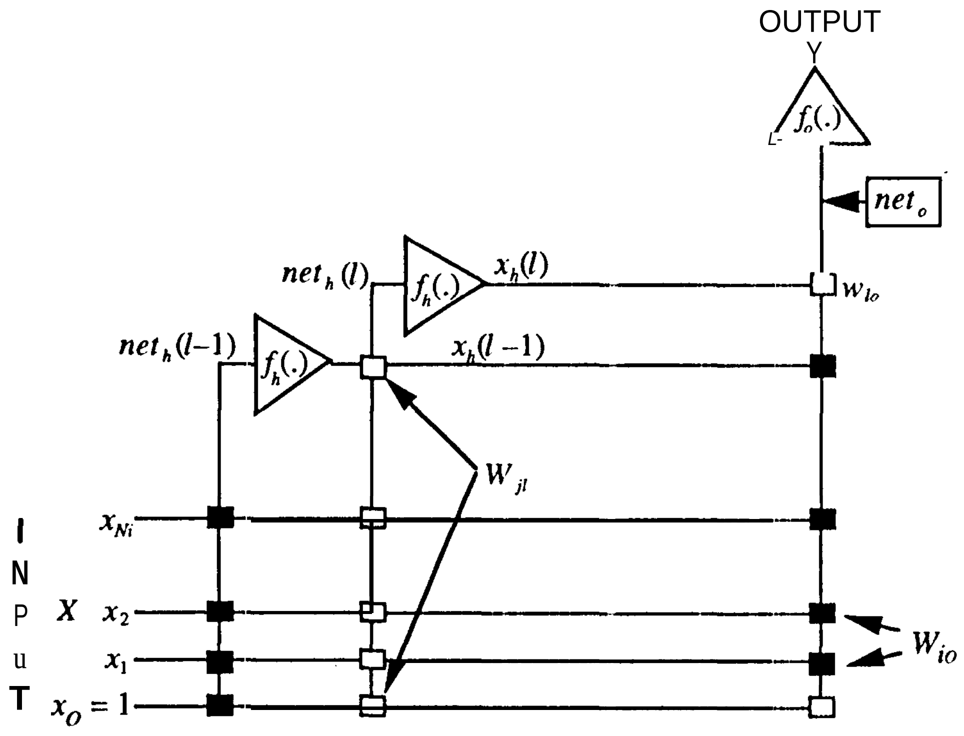
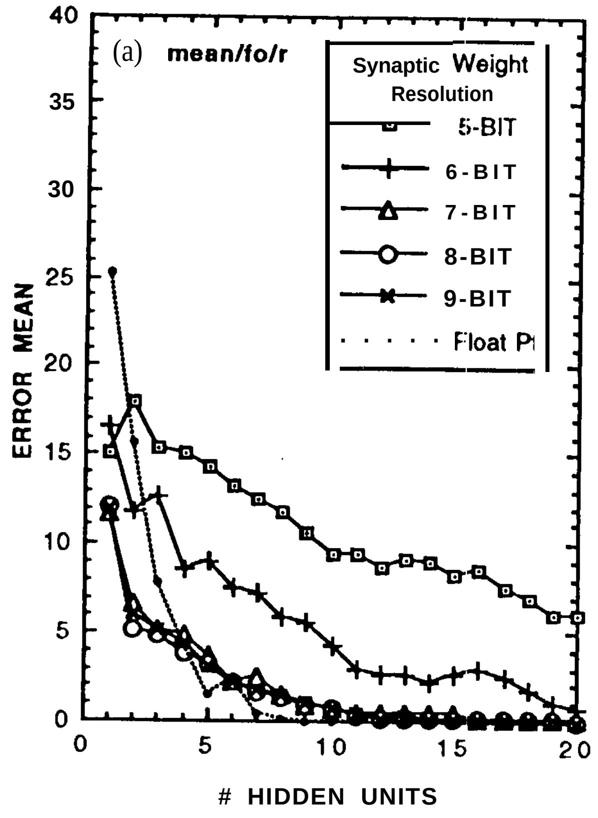
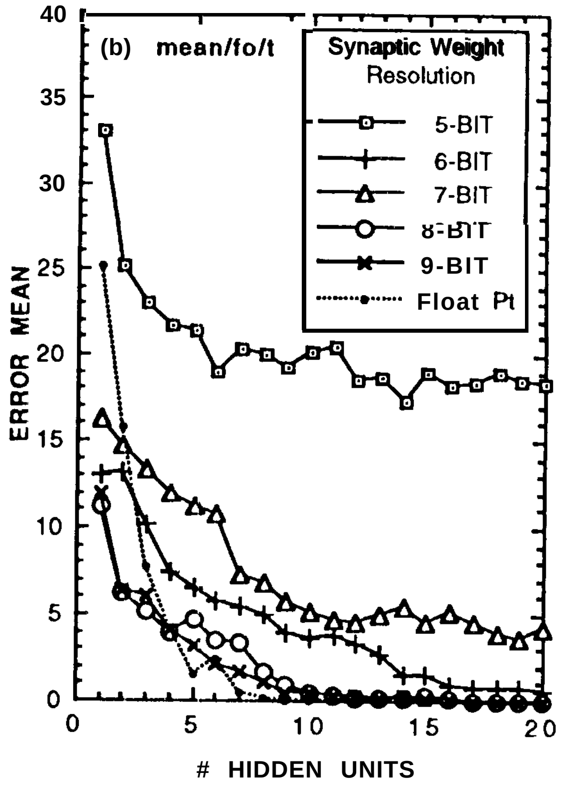

OUTPUT
Y
$f_o(.)$
net_o
w_{lo}
$net_h(l)$
$f_h(.)$
$x_h(l)$
$net_h(l-1)$
$f_h(.)$
$x_h(l-1)$
W_{jl}
I N P u T
X
x_{Ni}
x_2
x_1
$x_o = 1$
W_{io}

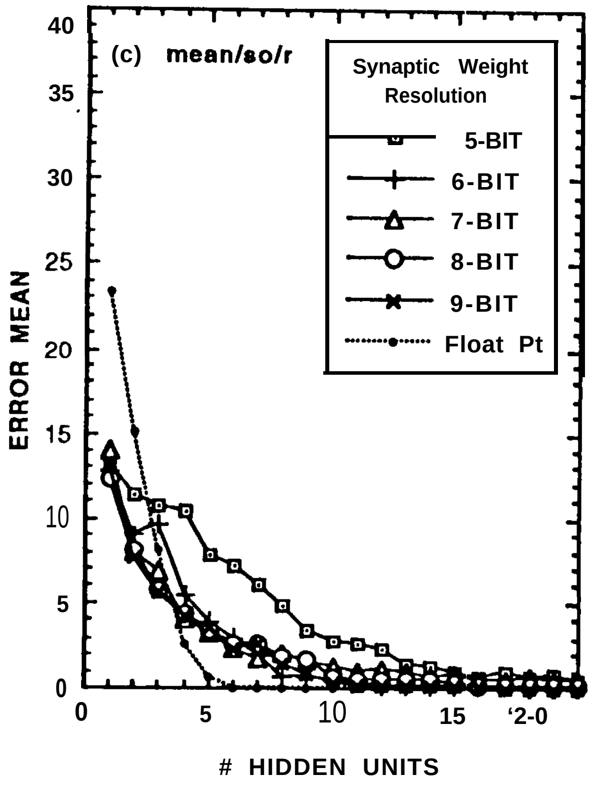
Figure 9



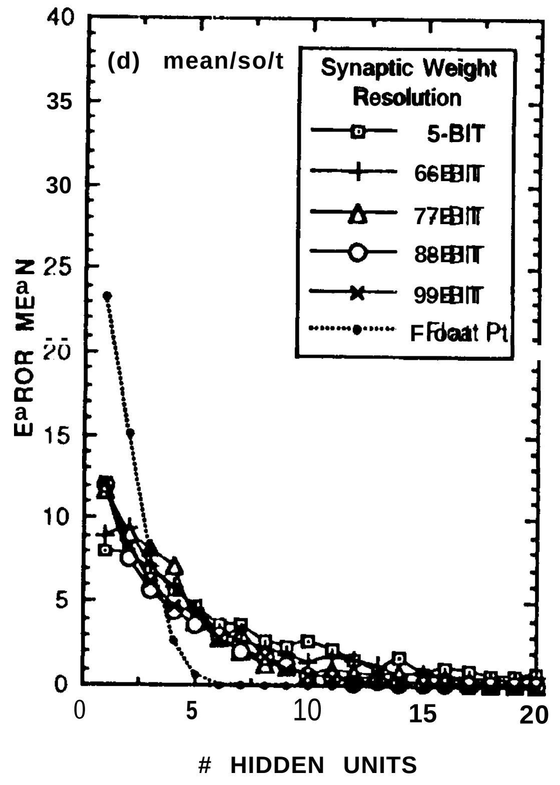
(a) mean/fo/r
Synaptic Weight Resolution
5-BIT
6-BIT
7-BIT
8-BIT
9-BIT
Float Pt
ERROR MEAN
# HIDDEN UNITS



(b) mean/fo/t
Synaptic Weight Resolution
5-BIT
6-BIT
7-BIT
8-BIT
9-BIT
Float Pt
ERROR MEAN
# HIDDEN UNITS

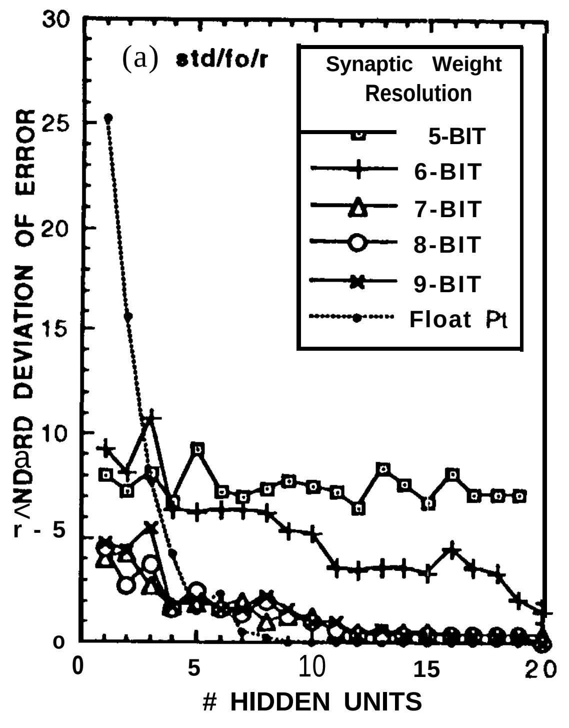


(c) mean/so/r
Synaptic Weight Resolution
5-BIT
6-BIT
7-BIT
8-BIT
9-BIT
Float Pt
ERROR MEAN
# HIDDEN UNITS

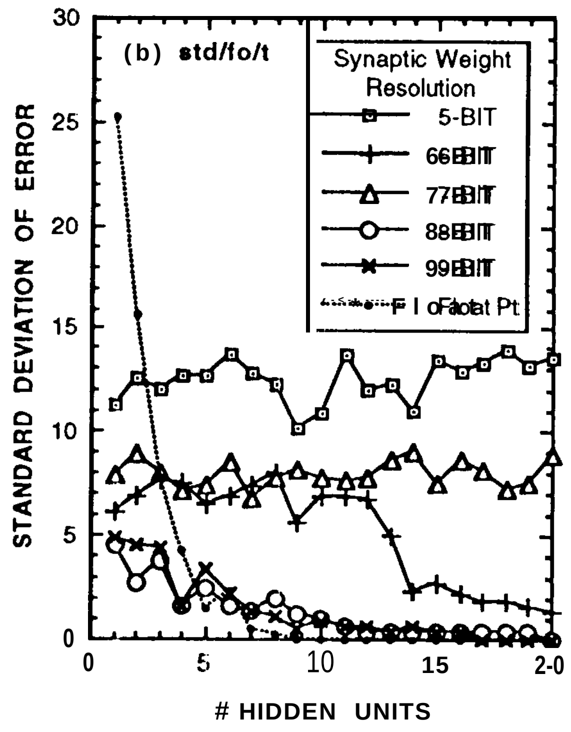


(d) mean/so/t
Synaptic Weight Resolution
5-BIT
6-BIT
7-BIT
8-BIT
9-BIT
Float Pt
ERROR MEAN
# HIDDEN UNITS

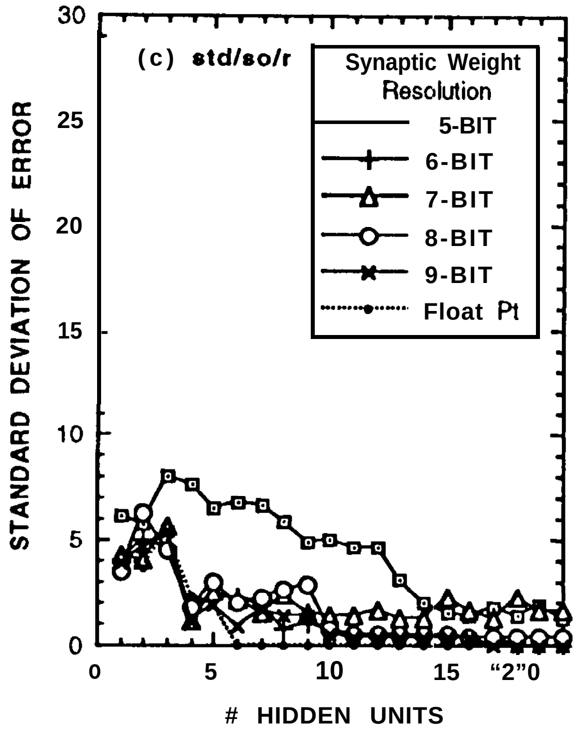
Figure 10



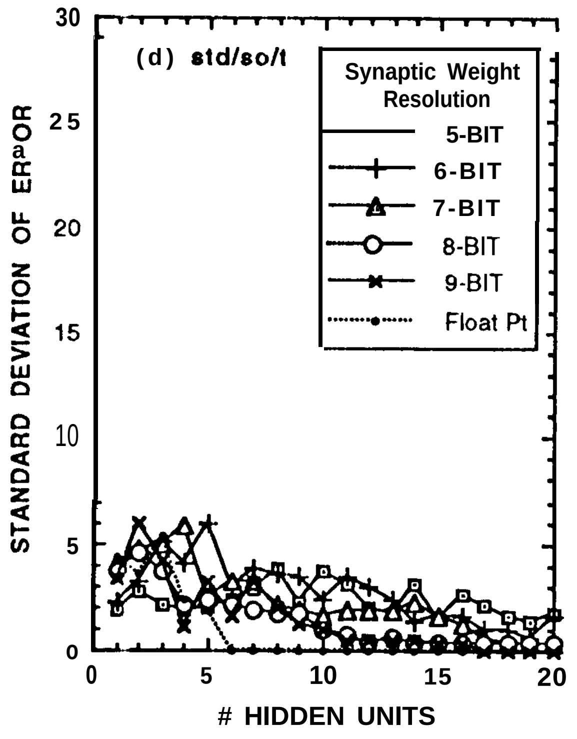
(a) std/fo/r
Synaptic Weight Resolution
5-BIT
6-BIT
7-BIT
8-BIT
9-BIT
Float Pt
STANDARD DEVIATION OF ERROR
# HIDDEN UNITS



(b) std/fo/t
Synaptic Weight Resolution
5-BIT
6-BIT
7-BIT
8-BIT
9-BIT
Float Pt
STANDARD DEVIATION OF ERROR
# HIDDEN UNITS



(c) std/so/r
Synaptic Weight Resolution
5-BIT
6-BIT
7-BIT
8-BIT
9-BIT
Float Pt
STANDARD DEVIATION OF ERROR
# HIDDEN UNITS



(d) std/so/t
Synaptic Weight Resolution
5-BIT
6-BIT
7-BIT
8-BIT
9-BIT
Float Pt
STANDARD DEVIATION OF ERROR
# HIDDEN UNITS

Figure 11

CASCADED VLSI 64X 64 **13-BIT** SYNAPTIC ARRAY AND 64 NEURONS NEURAL SYSTEM: HARDWARE LEARNING **FOR IMAGE RECOGNITION AND CLASSIFICATION**

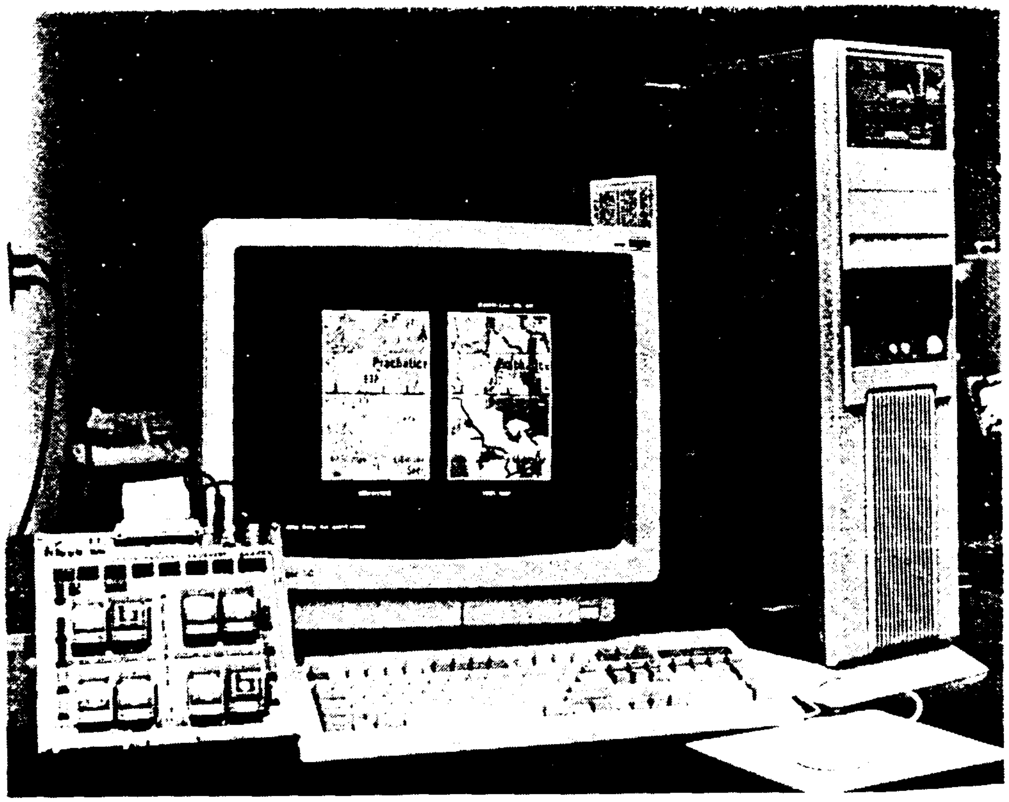
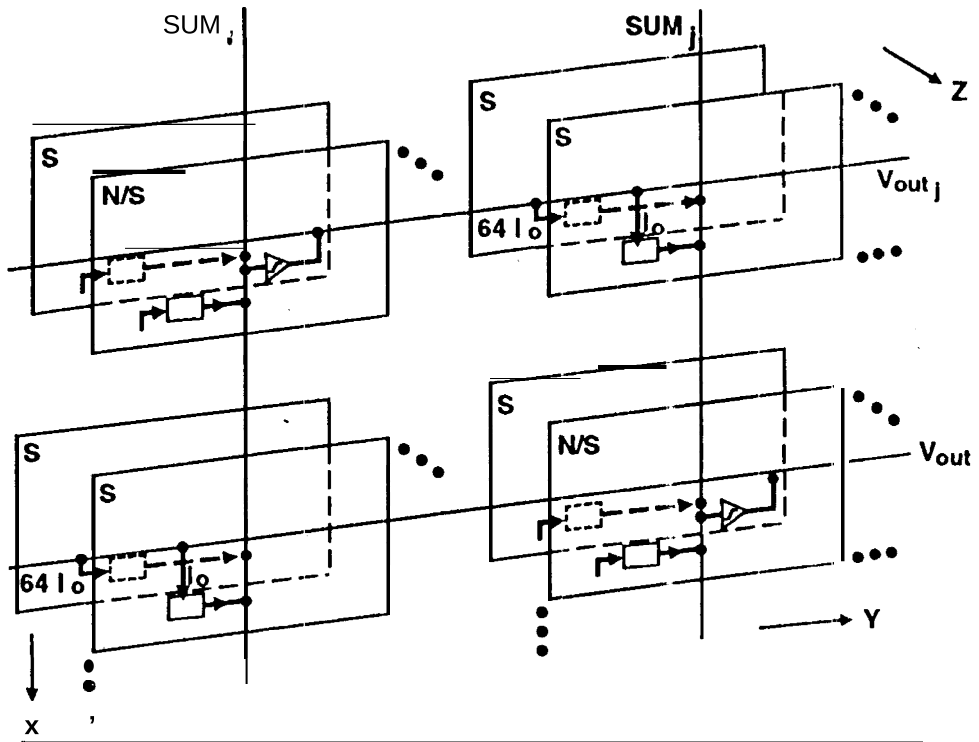


Figure 12



N/S NEURON-SYNAPSE CHIP S SYNAPSE CHIP X, Y DIRECTIONS FOR LARGER SIZE NETWORK Z DIRECTION FOR HIGHER SYNAPSE RESOLUTION	NEURON CELL SYNAPSE CELL

Figure 13

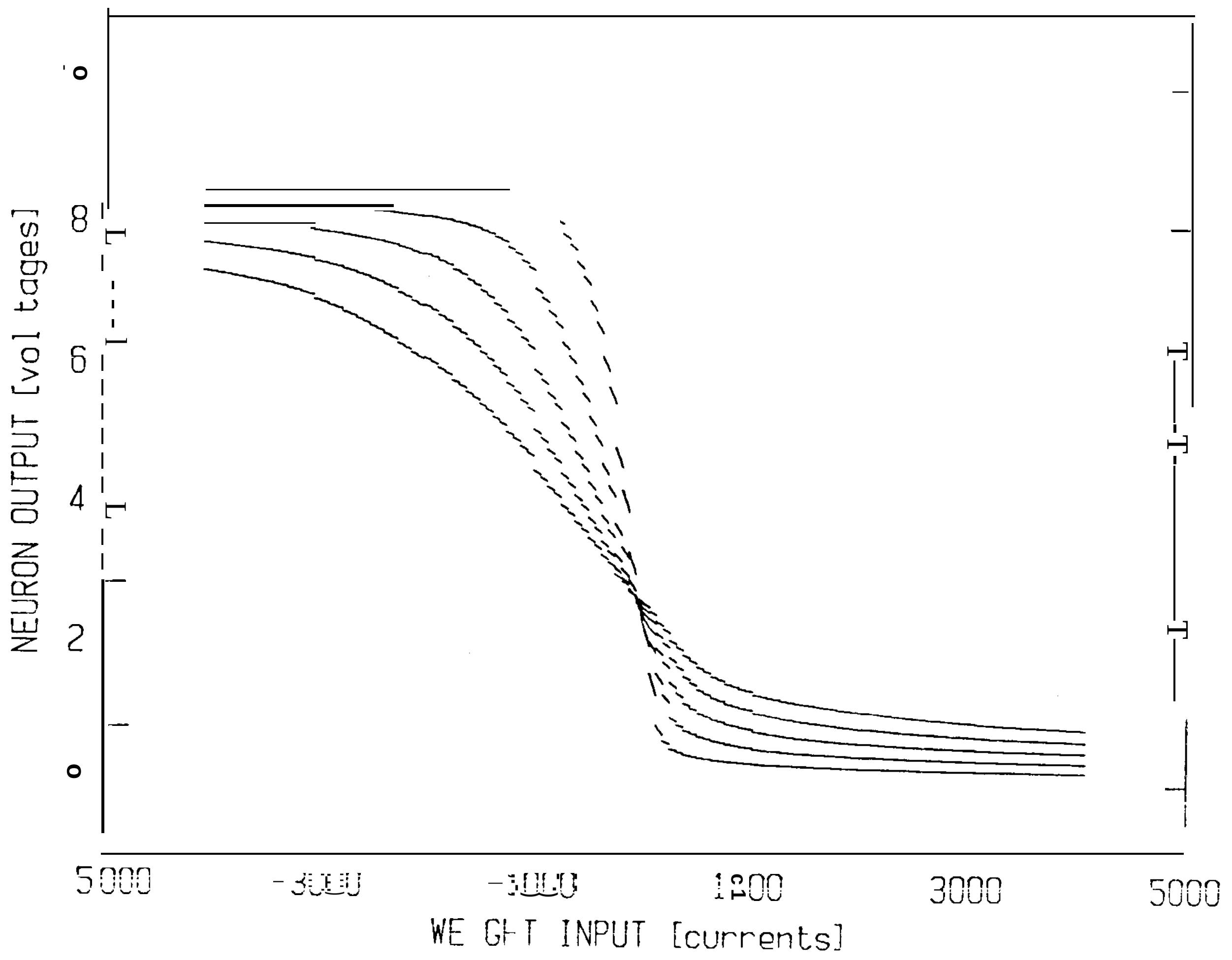

NEURON OUTPUT [vol tages]
o
8
6
4
2
o
5000
-3000
-1000
1000
3000
5000
WE GHT INPUT [currents]

Figure 14

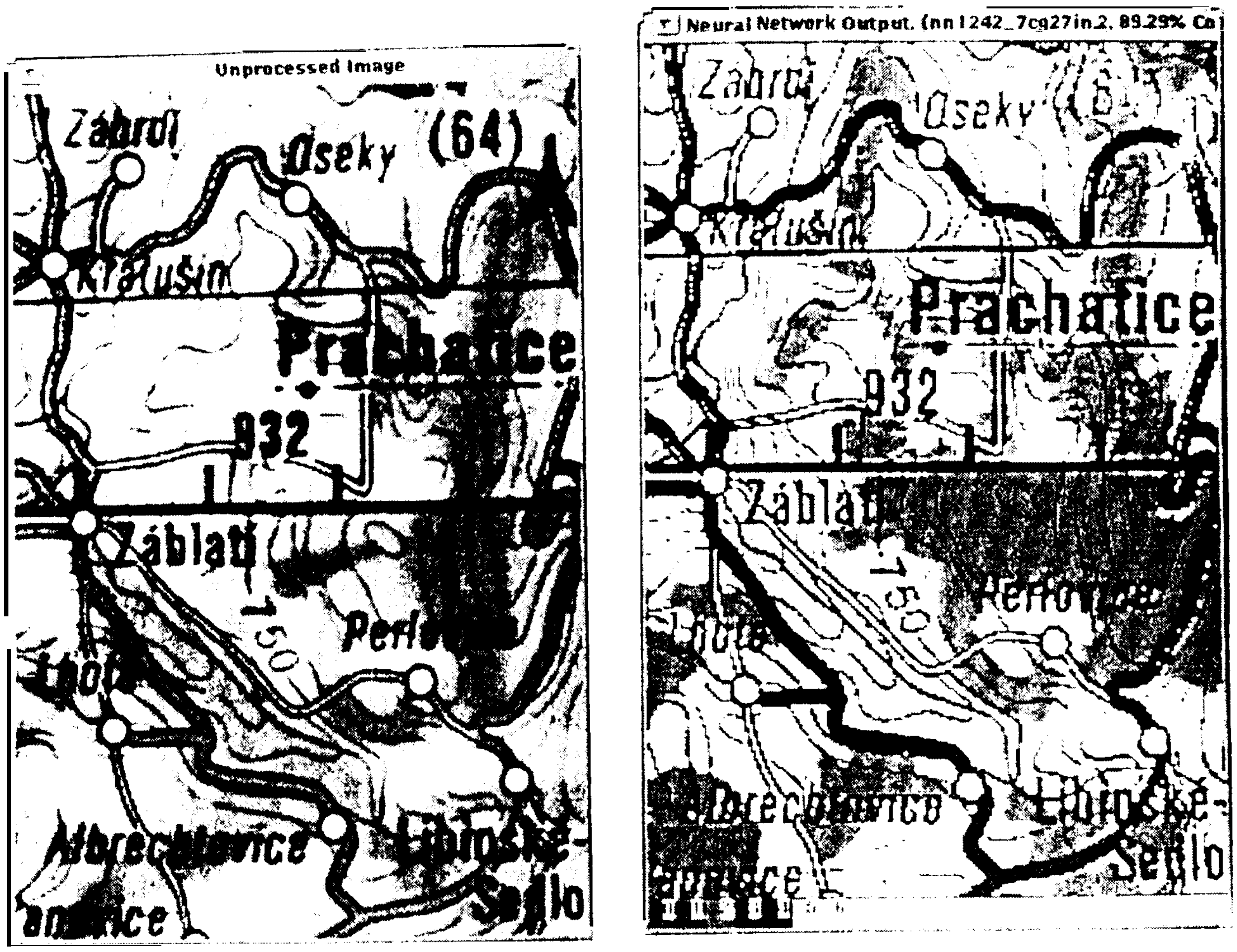

Unprocessed Image
Neural Network Output. (nn1242_7cg27in2, 89.29% Co
Zábrol
Oseky
(64)
Kratušín
Prachatice
932
Záblatí
150
Albrechtovice
Libínské
Sedlo

Figure 15

10/17/94 Section 1 of 5 W

TABLE 1 Seattle Office

Preliminary Survey of Neurally Inspired Chips and Boards

Electronic Neural Network Chips - Commercially available, under development, and/or imaginary

Company	Reference Number	Chip	First Silicon	Technology Source	Sponsor	Number of Transistors	Connection Updates/sec (cups)	Connections/sec (cps)	Semiconductor Technology	Clock Speed	Number of Neurons	Number of Weights	Algorithms Supported	Circuit Type	Partners	Miscellaneous
Adaptive Solutions, Inc. (Beaverton, OR)	NN22,NN26	1) CNAPS Prototype	12/90	Co-designer Inova MicroElec. (IME filed bankrupt in 1991)	Venture Funds		2.50E+08	1.60E+09	0.8u CMOS		64		BP		Mitsubishi Elec.,Sharp, Meidensha	(buggy)
	NN28	2) CNAPS-1064				1.40E+07	1.00E+08	1.28E+09	0.8u CMOS	20 MHz	64		LVQ2	Digital		Used in server with up to 512 processors
American Neurologix (Sanford, FL)		1)Prototype	1992												Samsung	Suspended neural work
		2) NLX-220														Fuzzy processor
AT&T Bell Labs (Holmdel, NJ)	NN11	1) ANNA			US Army SDC	180,000		1.00E+10	0.9u CMOS	20 MHz	8	4,096		Mixed A/D		For OCR
Bell Communications Research Corp., Inc. (Morristown, NJ)	NN46		In progress													
Boeing Defense & Space (Seattle, WA)		1) AVPS-POC2						2.50E+08	2u CMOS	10 MHz	40	960		Hybrid		Cascadable
CA Institue of Technology (Pasadena, CA)	NN77	1) Contour-Length			NSF, ONR, and RISC				2u CMOS		2,304			Analog	Tanner Research RISC	Dynamic wire tech. for vision circuitry
	NN77	2) Figure Ground							2u CMOS					Analog		
	NN79	3)			DAPRA/NSF				2u CMOS		6		EGD, K-W	Analog		
CalTech & JH Univ. (Pasadena, CA)	NN90				RIA-NSF, MOSIS,HP				2u CMOS n-well						C. Mead	
Center for Neural Engr. at USC (LA, CA)	NN07	1) Analog VLSI			DARPA,TRW, & Samsung				2u CMOS double polywell.	1.024 MHz	33	252	EKF	Analog		For communication receiver applications
CISC - Univ. of Michigan (Ann Arbor, MI)	NN06	1) 16 Channel CMOS			NIH	7,100			3u CMOS 1 metal/1 poly	10 MHz						80mW power dissipation
Computational NN Center (Lyngby, Denmark)	NN14	1) Neuron chip							CMOS		100	10,000		Analog		Arbitrary topology
Echelon Corporation (Palo Alto, CA)	NN18	1) MC143120													Motorola & Toshiba	External data bus
	NN18	2) MC143150														Self contained memory (EEPROM,SRAM,ROM)
Ecole Polytech/Fed. Laus (Lausanne, Switzerland)	NN60	1) L-Neuro 1.0				160,000		2.56E+07	1.6u CMOS		64	960	BP, Kohen., Hebbian			Real-time robotics applications
Fujitsu (Kawasaki, Japan)	NN31	1)	1988								50		BP	Digital		
	NN70	2)	In progress						2u BI-CMOS					Analog		Dispersion problems
General Dynamics (Pomona, CA)	NN23,NN32	1) AXON 1	1990					1.20E+09			40					Technology transferred to E-metrics(Ontario,CA)
Hitachi, Ltd. (Tokyo, Japan)	NN30	1) Prototype	1989						0.8u CMOS		576	1,152				
	NN17,NN27	2) 1024 Prototype	In progress	Central Research Laboratory				1.37E+09	0.5u CMOS	3.9 MHz	1,024	1.00E+06		Digital		75mW power dissipation
Hughes Research Lab (Malibu, CA)	NN74	1) 3D Wafer						2.4E+09 - 2.04E+10	3-D WSI							Vision integration net and backprop net
Hughes Tech. Center (Carlsbad, CA)	NN27		In progress					1.00E+07	EEPROM		300	100,000			NOSC (SanDiego)	
HNC (San Diego, CA)	NN26,NN28	1) SNAP	1992		US Army/ Ft. Monmouth	90,000 gates	6.25E+06	8.00E+07	1.0u CMOS gate array		4		BP/CP, SOM, LVQ2, ART,PNN, and others	Digital		Designed to work in 16 chip sets (1.28E+09 cps)
	NN26,NN28	2)VIP	1992		DARPA	110,000 gates	1.00E+08	3.2E+08 - 1.28E+09	1.0u CMOS gate array		64			Digital		Two chip set (VIP-1 and VIP-2)
Intel (Santa Clara, CA)	NN13,NN26, NN40,NN53	1) ETANN (80170NX)	1989	Naval Weapons Center			1.20E+09	2.00E+09			64	10,240	BP/RBP Madaline III			
	NN20,NN28 NN82	2) NI1000	2/93		DARPA/ONR	3.70E+06		1.00E+10	0.8u EEPROM	25 MHz	1,024	2.56E+05	RCE	Digital	Nestor	ATC applications
Irvine Sensors Corp. (Costa Mesa, CA	NN19,NN25, NN43	1) 3DANN			SDIO,ONR, IST,BMDO			3.40E+10		50 MHz	16,384			Analog		Human eye and brain emulator

W

Preliminary Survey of Neurally Inspired Chips and Boards

Electronic Neural Network Chips - Commercially ● bnllabls, under development, and/or imaginary (continued)

Company	Reference Number	Chip	First Silicon	Technology Source	Sponsor	Number of Transistors	Connection Updates/sec (cups)	Connections/sec (cps)	Semiconductor Technology	Clock Speed	Number of Neurons	Number of Weights	Algorithms Supported	Circuit Type	Partners	Miscellaneous
JPL (Pasadena, CA)	NN01,NN84	1) Path Planner						6.00E+09	2u CMOS n-well	7 MHz	600			Digital		
	NN02,NN03	2) Neuron-Synapse Building Block						5.00E+08	2u CMOS p-well		32	992	Cascade BP	Analog		Currently being commercialized
	NN47	3) Asset Manager							2u CMOS		1,600			Analog		
		4) Associative Memory							2u CMOS				Cascade BP	Hybrid		Stores 128 patterns of dimension 16
King's College (London, UK)	NN04	1) pRAM			SERC	30,000 gates			1u CMOS		256					Up to 5 chips can be linked
Korea Telecom (Seoul, Korea)		1) URAN						200E+09	1.2u CMOS		14-225	3,598		Digital		
Matsushita Electric (Osaka, Japan)		1) QNC				27,000 gates	500E+c4	2.05E+10	1.2u double metal CMOS		4,736	2.00E+06				Weights are stored off chip
Melco-Mitsubishi Electric (Hyogo, k - I	NN08	1) NN LSI Circuit		LSI Laboratories				8.00E+04	0.8u CMOS		400	40,000	Boltzmann	Analog		
Micro Computing Lab (Lausanne, Switzerland)	NN16	I) OWES HN8	19s0			380,000		1.10E+08		10 MHz	16		Hopfield		SFIT	4x4 chip set
	NN15	2) GENES HH8	1991			800,000		1.00E+09		40 MHz	24		Hebbian	Digital		
Micro Devices (Lake Mary, FL)	NN38,NN53	1) MD 1220 (Neural Bit Slice)						1.00E+07			8			Digital		Weights stored in external memory
MIT Lincoln Laboratory (Lexington, MA)	NN91	1) NNC2						1.92E+09	2u CCD/CMOS dm-dps	10 MHz		980		Digital		Tecn. based on charge coupled device
Neural Semiconductor (Irvine, CA)	NN38,NN53	1) CNU3232S	1992							25 MHz	32	1,024		Digital		$500/chip 100,000 patterns/sec
Neural Technologies, Ltd. (Petersfield, England)	NN41,NN46	1) NISP													Smith Industries	
North Carolina University (Raleigh, NC)	NN12,NN34	1) TInMANN			IBM	4,000		1.96E+05	2u CMOS p-well	15 MHz	1		Kohonen, Markovian	Digital		
NTT LSI Laboratories (Kanagawa, Japan)	NN09	1) PDN Model				15,500 gates 582,400 trans.		8.00E+09	0.8u CMOS	5.6 MHz	13	832		Digital		Low-power chain-reaction (LCR) arch.
Oxford Computer, Inc. (Oxford, CN)	NN42	1) OBL Chip							Submicron CMOS				unrestricted	!		
Ricoh, Ltd. (Tokyo, Japan)	NN23	1) RN-100	4/90				4.00E+07	8.00E+07		10 MHz	1	8				
	NN23	2) RN-200	7/92			200,000 gates	1.50E+09	3.00E+09	0.8u CMOS	12 MHz	16	256		Digital		To be used in photocopier
Siemens, AG (Munich, Germany)	NN21	1) MA-16 Prototype	1992		Esprit	610,000 gates	4.00E+08	4.00E+08	1.0u CMOS	25 MHz (demo)	128	65,536	unrestricted	Digital		$2K/chip
	NN61,NN65, NN67,NN83	2) MA-16				510,000		8.00E+08	1.0u CMOS	50 MHz	16+	16+		Digital		Signal preprocessing
Stanford University (Stanford, CA)	NN81				NASA	400,000	3.50E+08		1.2u CMOS	125 MHz	32	20,480	Boltzmann	Digital		
Sydney Univ. EE (Sydney, Australia)	NN59	1) Kakadu												Analog		20uW power conuption
	NN80	2) WATTLE							1.2u CMOS		10	84	CSA	Analog		
Synaptics, Inc. (San Jose, CA)	NN42,NN69	1) I-1000 Neural Eye												Analog		<100mW
	NN76	2) RBF Chip			DITRD				2u CMOS	90kHz				Analog		For Radial Basis Func.
Tohoku Univ. (Sendai, Japan)	NN73	1) Neuron MOS Transistor (vMOS)							Double poly-silicon CMOS							Logic design using Floating-Gate PD
Toshiba (Tokyo, Japan)	NN24,NN75	1) Synapse Chip				150,000	1.80E+09		0.8u double metal CMOS	50 MHz		576	Amari-Hopfield	Analog		Layered Neural Net Two chip set (neuron & synapse chip work together)
	NN24,NN75	2) Neuron Chip				11,000			0.8u dm CMOS		24		BP, Hebbian	Analog		
Toyoshi Univ. of Tech. (Toyoshi, Japan)	NN64												BP			Based on optoelectronic integrated circuit model

Preliminary Survey of Neurally Inspired Chips and Boards

Electronic Neural Network Chips - Commercially available, under development, and/or imaginery (continued)

Company	Reference Number	Chip	First Silicon	Technology Source	Sponsor	Number of Transistors	Connection Updates/sec (cups)	Connections/sec (cps)	Semiconductor Technology	Clock Speed	Number of Neurons	Number of Weights	Algorithms Supported	Circuit Type	Partners	Miscellaneous
TRW (San Diego, CA)	NN29		In progress						1.0u CMOS							
Univ. of California & ICSI (Berkeley, CA)	NN16	1) SPERT	In progress		ONR & NSF		1.00E+08	1.00E+11	MOSIS CMOS	50 MHz		64	BP			
Univ. of Delaware (Newark, DE)	NN86													Analog		VLSI neuromorph
Univ. of Edinburgh (Edinburgh,)	NN78	1) EPSILON						3.60E+08	1.5u CMOS dmsp			3,600		Analog		Pulse stream neural state signalling
Univ. of Minnesota (Duluth, MN)	NN58								2.0u CMOS			9	BEP	Analog		LSI prototype
Univ. of Southern Calif. (Los Angeles, CA)	NN63 NN68	1) Early Vision 2) Video Motion Detector						1.80E+10 8.32E+10	1.2u CMOS 1.2u CMOS		5 25	 875		Analog Analog		Multiple neuro-processor chip
Univ. of Tsukuba (Tsukuba, Japan)	NN62	1) Pulse Density Modulating (PDM) Chip												Digital		High Scalability
Univ. of Waterloo (Waterloo, ONT, CAN)	NN85								0.8u BICMOS					Mixed A/D		
Univ. of WI-Platteville (Platteville, WI)	NN57	1) Neural Network Controller	1992						2.0u CMOS		17	15		Analog		Control applications
Washington St. Univ. (Pullman, WA)	NN87	1) PWTA											Hopfield			
Westinghouse (Baltimore, MD)	NN27,NN33	1) MSPSE	1993	Elec. Design, Inc. Analog Devices	ARPA				0.8u CMOS	33 MHz				Digital		To be used in RAH-66

Opto-electronic Neural Network Chips - Commercially available, under development, and/or imaginery

Company	Reference Number	Chip	First Silicon	Technology Source	Sponsor	Number of Transistors	Connection Updates/sec	Connections/sec	Semiconductor Technology	Clock Speed	Number of Neurons	Number of Weights	Algorithms Supported	Circuit Type	Partners	Miscellaneous
JILA - University of CO (Boulder, CO)	NN88	1) Photorefractive Ring Resonator			ONR & AFOSR								Lotka-Volterra			
JPL (Pasadena, CA)	NN50	1) Optoelectronic System						1.00E+12						Analog		Robotic vision and pattern recognition
Mitsubishi Electric corp. (Hyogo, Japan)	NN05,NN10 NN66	1) Optical Neural Chip	1988	Central Research Laboratory				1.00E+12		10 MHz	32	1024	BP	Analog		Neuron density is 2000/sq.cm
Tel-Aviv Univ. (Tel-Aviv, Israel)	NN71															Four quadrant matrix-vector multiplier

W
Seattle Office

Preliminary survey of Neurally Inspired Chips and Boards

Neural Network Boards - Commercially available, under development, and/or imaginary																
Accurate Automation Chattanooga, TN	NN51	1) Sparse MIMD AAC NNP						1.40E+08		33 MHz	8,000	32,000		Hybrid		Single/multi-processor environment
Adaptive Solutions, Inc. (Beaverton, OR)	NN43 NN55	1) CNAPS/VME16 2) CNAPS/PC Board	1994		ARPA					20 MHz	128					16 Processors 1.3 BOPS 6.4 BOPS
AND America, Ltd. (Ont., Canada)	NN36,NN62 NN36	1) HNet Applications Development System 2) HNet Supercomputing Expansion Board						64,000		25 MHz						ISA-One transputer ISA Bus - 8 transputers
AT&T Bell Labs (Holmdel, NJ)	NN89	1) ANNA Board														Uses ANNA NN chip & DSP32C processor
California Sci Software (Nevada City, CA)	NN28,NN36, NN52	1) BrainMaker Accelerator						3.00E+06		20 MHz			BP			TMS 320C25 - $2000
Current Technology, Inc. (Durham, NH)	NN37,NN64	1) MM32K-AT						5.00E+09		12.5 MHz	32,768					AT Bus - $5000
Intel (Santa Clara, CA)	NN38,NN39	1) INNTS								25 MHz	2,048	20,480				ISA Bus - $12,000
Micro Computing Lab (Lausanne, Switzerland)	NN15	1) GENES SY1						1.10E+08			1,728	288	Hopfield			
Mosaic Industries, Inc. (Newark, CA)	NN38	1) QED Board												Digital		
Neural Semiconductor (Irvine, CA)	NN38	1) Virtual Support Board	1992													4 CNU 3232S chips $7,500
Neurodynamx (Boulder, CO)	NN28,NN44, NN58 NN28,NN44	1) NDX XP50(b) 2) NDX XP25(b)					1.50E+07 7.50E+06	4.50E+07 2.25E+07		50 MHz 25 MHz			BP/RBP BP/RBP			$8,995
Oxford Computer, Inc. (Oxford, CN)	NN42	1) IPRMM														Matrix vector multiplier
Toshiba Corp. (Fuchu-shi, Japan)	NN72	1) MULTINEURO				100,000 gates per 2 nodes		1.50E+09	0.8 u CMOS		2.09E+06		MP,LVQ,BP Hopp, others	Digital		65 VLSI chip board
Ward Systems Group (Frederick, MD)	NN28	1) NeuroBoard								50 MHz			BP/SOM			ISA Bus - $1500-3000
Biological Systems - In production																
Mother Nature	NN35,NN48 NN49	1) Mk I - Human 2) Aplysia (Sea Snail)						>1.00E+15		~100 Hz	1.00E+10 1.00E+06	1.00E+14				

W

Reference List: Artificial Neural Network Chips & Boards

Ref. Num.	Article Title	Reference Title/Organization	Author	ite
NNo1	Rectangular Array of Digital Processors for Planning Paths	JPL New Tech Report # NPO-18727	Ewing, T.L.	Dec-93
N02	Cascaded VLSI Chips Help Neural Network To Loam	JPL New Tech Report # NPO-18645	Ewing, T.L	Dec-93
N03	Non-Volatile Army of Synapses for Neural Network	JPL New Tech Report # NPO-18578	Ewing, T.L	Dec-93
N04	The PRAM An Adaptive VLSI Chip	IEEE Transactions on Neural Networks	Clarkson, T.G	May-93
N05	Optical Learning Neurochip wkh Internal Analog Memory	Applied Optics	Nitta, Y	Mar-93
N06	A 16-Channel CMOS Neural Stimulating Army	IEEE Journal d Solid-State Circuits	Tanghe, SJ	Dec-92
	A Programmable Analog VLSI NN Processor for Communication Receivers	IEEE Transactions on Neural Networks	Choi, J	May-93
N08	A Refreshable Analog VLSI NN Chip with 400 Neurons ● nd 40K Synapses	IEEE Journal d Solid-State Circuits	Arima, Y.	Dec-92
N09	A High-Speed Digital NN Chip wkh Low-Power Chain-Reaction Architecture	IEEE Journal d Solid-State Circuits	Uchimura, K	Dec-92
	Melco's Neural Chip; Holography Research	IEEE Micro	Kahaner, D.K	Aug-92
	Application of the ANNA NN Chip to High-Speed Character Recognition	IEEE Transactions on Neural Networks	Sackinger, E	May-92
	The TInMANN VLSI Chip	IEEE Transactions on Neural Networks	Melton, M S	May-92
	The MOO 2 Neurocomputer System Design	IEEE Transactions on Neural Networks	Mumford, M L	May-92
	An Analog CMOS Chip Set for NNs with Arbitrary Topologies	IEEE Transactions on Neural Networks	Lanener, J A.	May-93
	A Generic Systolic Array Building Block for NNs with On-Chip Learning	IEEE Transactions on Neural Networks	Lehmannn, C	May-93
	The Design of ● Neuro-Microprocessor	IEEE Transactions on Neural Networks	Wawrzynek, J	May-93
	A Single 1.SV Digital Chip for ● 10^6 Synapse Neural Network	IEEE Transactions on Neural Networks	Watanabe, T.	May-93
	Echelon: Networking Control	IEE Review	Heath, S	Oct-92
	Neural Network System to Stimulate Human Brain ● nd Eye	Defense Electronics	K., L.A.	Ott-93
N20	Darpa Gets Neural Chip from Intel	Electronic Engineering Times	Johnson, RC.	Feb-93
	Siemens Fields Big, Fast Neural IC	Electronic Engineering Times	Johnson, R.C	Dec-92
N22	Adaptive Adapts its CNAPS Chip	Electronic Engineering Times	Wirbel, L	Mar-93
N23	Ricoh Announces Neural Microchip	Electronic Engineering Times	Yoshida, J	Jul-92
	Neural IC Beefed Up	ElectronEngineering Times		Sep-92
	SD Stacking Used to Build Neural Network	Electronic Engineering Times	Wirbel, L	Sep-92
N26	Neural Chips Pour 10DN 'cups'	Electronic Engineering Times	Johnson, RC	Oct-92
	Neural Microchips Go Commercial	Electronic Engineering 1 imes	Johnson, R.C	Jun-92
N28	Working with Neural Networks	IEEE Spectrum	Hammerstrom, D	Jul-93
N29	Raytheon Dealing for TRW's IC Business	Electronic Engineering Times	Gold, M	Jun-92
N30	Hitachi Neural Prototype Thinks, Learns Quickly		Reinhardt, A	
	Fujitsu Plays Cops 'N Robbers to Apprehend Expert AI	Electronic Engineering Times	Johnson, R.C	Feb-89
	GD Claims Neural Chii Breakthrough	Military & Aerospace Electronics	Adams, C.	
N33	Westinghouse Readies Neural Network Computer	Military & Aerospace Electronics	Keller, J	Apr-93
	TInMANN The Interger Markovian Artificial Neural Network	Computer Science Laboratory	van den Bout, D.E	Jul-89
N35	Electronic Neural Network Chips	Applied Optics	Jackel, L D	Dec-87
N36	Neural Network Resource Guide	AI Expert	Shaw, J.	Feb-93
	Introducing the MM32K-AT	Current Technology, Inc	Current Tech	
N38	MD1220 - Neural Bit Slice	Micro Devices	Internal DL	Feb-94
N39	Neural Network Solutions	Intel	Intel	
N40	80170NX - ETANN	hdd	Intel	Mar-23
	PCs Get Neural-Net Training	Electronic Engineering 1 -	Woolnough, R	Aug-92
	Are Artificial Neural Networks Finally Ready for Market?	Electronics	Manuel, T.	Aug-88
	Neural Nets Carve ● Niche in Military Systems	Military & Aerospace Electronics	Keller, J	Feb-94
	Hardware Accelerator Card for Neural Net Training	PC AI		Dec-93
	In Britain, Neural Nero ● re Bursting Out All Over	Electronic Engineering Times	Johnson, R C	Nov-92
N46	Computers that Loam by Doing	Fortune	Bylinsky, G	Sep-93
	A Neuroprocessor for Asset Management	JPL, Calif Institute of Technology	Daud, T.	
	The Retina An Approachable Part d the Brain	Belknap/Harvard Univ. Press	Dowling, J E	19s7
	Neurons ● ld Networks An Introduction to Neuroscience	Belknap/Harvard Univ. Press	Dowling, J E	1992
N50	Fast Feature-Recognizing Optoelectronic System	NASA Tech Briefs	Thakoor, S	Nov-90
	Sparse MIMD Neural Network Processor	Accurate Automation Corp.	M C	1993
	Neural Networks	PC AI		May-94
N53	Advanced Methods in Neural Computing	Von Nostrand Reinhold	Wasserman, P.D	1993
N54	A Massively-Parallel SIMD Processor for Neural Network/Machine Vision	Current Technology, Inc	Glover, MA	
N55	ARPA Awards $2 Million for Massively-Parallel PC	Military & Aerospace Electronics	Rayner, S	Feb-94
N56	Product Showcase: Accelerate Your Train	AI Expert		Jul-93
	On the Design of a Neural Network Chip for Control Applications	Proceedings - IEEE Inter.Symp C&S	Narathong .	1993
N58	Modular Analog CMOS LSI for Feedfrwd NN with On-Chip BEP Learning	Proceedings - IEEE Inter.Symp C&S	Wang, Y	1293
N59	Low Power Trainable Analog Neural Netwrok Classifier Chip	Proceedings - IEEE CIC Conference	Leong, P.	1993
N60	Neural Accelerator for Parallelization d Back-Propagation Algorithm	Microprocessing ● nd Microprgramming	F ranzi, E	Sep-93
	Architecture ● nd VLSI Design of a VLSI Neural Signal Processor	Proceedings - IEEE Inter Symp C&S	Ramacher, u	1993
N62	PDM Digital Neural Network Sytem	Electronics h Communications in Japan	Hirai, Y.	Feb-92
N63	VLSI Neuroprocessor for Image Restoration	Journal d VLSI Signal Processing	Lee, J-C	Apr-93
N64	Optoelectronic Adaptive Device and Its Learning Performance	Electronics & Communications in Japan	Kanamori, K	Nov-92
N65	General-Purpose Signal Processor Architecture for Neurocomputing	Journal of VLSI Signal Processing	Ramacher, u	Jun-93
N66	Optical Neuro-Devices	Optoelectronics - Devices& Tech.	Kyuma, K	Mar-93
	Design of ● General-Purpose Neural Signal Processor	Neurocomputing	Beichter, J	Feb-93
N68	VLSI Neuroprocessors for Video Motion Detection	IEEE Transactions on Neural Networks	Lot, J-C	Mar-93
N69	Neural Nets ● m Starting to Make Sense	Biosensors & Bioelectronics	McDonald, JA.	Sep-92
	Analog Neurochip ● nd Its Applications 60 Multilayered Artificial NN	Fujitsu Scientific and Tech Journal	Masumoto, D.	Fall 93
	Four-quadrant Optical Matrix Vector Multiplication Machine as ● NNP	Proceedings of SPIE	Abramson, S	1993
	Programmable Parallel Digital Neurocomputer	IEICE Transactions on Electronics	Shimokawa, Y.	Jul-93
	Neuron MOS Binary-Logic Integrated Circuits - Part I	IEEE Transactions on Electron Devices	Shibata, T.	Mar-23
	3.0 Wafer Stack Neurocomputing	IEEE Inter Conf on Wafer Scale Int	Campbell, M L	1993
	Neuro Chips wkh On-Chip BP tier Hebbian Learning	IEEE Journal d Solid-State Circuits	Takeshi, S	Dec-93
	An Analog VLSI Chip for Radial Basis Functions	ANIPS 5	Platt, J.C	1993
	Object Based Analog VLSI Vision Circuits	ANIPS s	Koch, C	1993
	Generic Analog Neural Computation - The EPSILON Chip	ANIPS s	Churcher, S	1993
	A Learning Analog NN Chip wih Continuous-Time Recurrent Dynamics	ANIPS 6	Cauwenberghs, G	1994
N80	WATTLE: A Trainable Gain Analogue VLSI Neural Network	ANIPS 6	Coggins, R	1294
	Digital Boltzmann VLSI for Constraint Satisfaction and Learning	ANIPS 6	Peterson, A M	1994
	Intel and Nestor to Commercialize Neural-Net Chip	BYTE	Bara, N	Mar-94
N83	Special Chips ● m at the Core of World's Fastest Neuro-Computer	Electronics	Gosch, J	Jan-93
N84	High-Speed Path Planning	JPL	Kemeny, S.	1204
N85	An Improved Programmable NN and VLSI Architecture Using BiCMOS	Proceedings - 1S94 Int NN Society AM	Zhang, D.	Jurl-94
N86	Analog VLSI Neuromorph with Spatially Extensive Dendritic Tree	Proceedings - 1994 Int NN Society AM	Elias, J G	Jun-94
	VLSI Implementation of ● Pulse-Coded Winner-Take-All Network	Proceedings -1024 Int NN Society AM	Meador, J L	Jun-M
N88	Optical Implementation d ● Self-Organizing Feature Extractor	ANIPS 4	Anderson, D.Z.	1293
N89	A Neurocomputer Bored Based on the ANNA Neural Network Chip	ANIPS 4	Sackinger, E	1992
N90	A Contrast Sensitive Silicon Retina with Reciprocal Synapses	A N I P S 4	Boahen, K.A.	1292
	CCD Neural Network Processors for Pattern Recognition	ANIPS 4	Chiang, A.M	1992

W